

AD-A274 218



①

AFIT/GE/ENG/93S-37

S DTIC
ELECTE
DEC 23 1993
A

DEVELOPMENT OF A
PERFORMANCE EVALUATION TOOL (MMSOFE)
FOR DETECTION OF FAILURES WITH
MULTIPLE MODEL ADAPTIVE ESTIMATION (MMAE)

THESIS

Robert L. Nielsen
Captain, USAF

AFIT/GE/ENG/93S-37

93-31062

180 P

Approved for public release; distribution unlimited

93 12 22 1 75

AFIT/GE/ENG/93S-37

DEVELOPMENT OF A PERFORMANCE EVALUATION
TOOL (MMSOFE) FOR DETECTION OF FAILURES
WITH MULTIPLE MODEL ADAPTIVE ESTIMATION (MMAE)

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Robert L. Nielsen, B.S. Physics, B.S. Aeronautical Engineering
*
Captain, USAF

December 1993

DTIC QUALITY INSPECTED 3

Approved for public release; distribution unlimited

Accession For	
NTIS	CRASH <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Preface

The product of this thesis (MMSOFE) should prove to be a useful software tool to the US Air Force guidance and control community and a valuable starting point for developing other multiple model application software. MMSOFE could prove to be either a boon or a bane to future AFIT guidance and control students who may be required to use MMSOFE to perform multiple model adaptive estimation with N elemental Kalman filters instead of just a simple single Kalman filter simulation like students in previous years. To those future students, I say, "You're welcome." While originally intended as a preliminary software development portion of this thesis to be used on a larger more complex problem, the development of MMSOFE itself proved to be easily thesis-worthy and challenging.

First and foremost, neither the successful completion of this thesis nor my enduring the entire AFIT experience would have been possible without the excellent support of my loving wife, Vera, and my dear children; Lynsa, Eric, Amber, and Lauren (15, 12, 9, and 5 years). Their unselfish sacrifice of time with me and of my attention speaks magnitudes for their characters and for their continuing loving and prayerful support. Such cannot be repaid, nor should the sacrifice of any AFIT student's family be taken for granted. To my family, "I love you, and thank you for enduring with me."

My gratitude especially goes to Dr. Peter Maybeck for his painstaking proofreading, for his advice and direction, for being a touchstone for sanity-checking, and most especially for his continuing prayers of support. Additionally, LtCol Robert Riggins deserves a special "thank you" for discussing software bugs and development problems and for his excellent comments and corrections to this thesis. My appreciation is also due Maj Randall Paschall for taking time in his overfull schedule to proofread and to those at Wright Laboratory for bearing with me while I finished this effort. Last, but not least, "Thank you" to Mr Donald E. Smith of Co-Operative Engineering who spent many hours ensuring adequate disk space was available and for keeping "MY" SPARC operating.

p.s. Never again!

Robert L. Nielsen

Table of Contents

	Page
Preface	ii
Table of Contents	iii
List of Figures	vii
List of Tables	x
List of Symbols	xi
Abstract	xiii
I. Introduction	1-1
1.1 Background	1-2
1.2 Problem Definition	1-4
1.3 Scope	1-5
1.4 Assumptions	1-5
1.5 Literature Review	1-6
1.5.1 Other Error Detection Methods.	1-7
1.5.2 Multiple Model Adaptive Estimation (MMAE).	1-10
1.5.3 MMAE Applications.	1-12
1.5.4 Literature Review Conclusion.	1-12
1.6 Methodology	1-13
1.6.1 MMAE Implementation Options.	1-13
1.6.2 Software Development.	1-14
1.6.3 Preliminary Studies.	1-15
1.6.4 Software Validation Tests.	1-16

	Page
1.6.5 Follow-on Goals.	1-17
1.6.6 Stopping Criteria.	1-18
1.7 Overview of Thesis	1-18
 II. Kalman Filtering, MMAE, and Failure Detection	 2-1
2.1 Overview	2-1
2.2 The Extended Kalman Filter	2-1
2.2.1 The Sampled Data Kalman Filter.	2-1
2.3 Multiple Model Adaptive Estimation	2-5
2.4 Failure Detection	2-9
2.4.1 FDI via MMAE.	2-9
2.4.2 Threshold Selection and Filter Tuning.	2-9
2.5 Stochastic Adaptive Control	2-11
2.6 Distributed Kalman Filtering	2-12
2.7 Summary	2-13
 III. MMSOFE Development	 3-1
3.1 Overview	3-1
3.2 Satellite Orbit Problem	3-1
3.3 MSOFE Description	3-2
3.4 MMSOFE Adaptation from MSOFE	3-13
3.4.1 Input/Output Files.	3-13
3.4.2 New Subroutines.	3-16
3.4.3 Versions of MSOFE Subroutines.	3-20
3.4.4 Vectorization and Matricization.	3-28
3.5 MMSOFE Validation	3-35
3.6 Summary	3-37

	Page
IV. Analysis of Results	4-1
4.1 Overview	4-1
4.2 Elemental Filter Description	4-3
4.3 Probabilities	4-3
4.3.1 Expected Probabilities.	4-3
4.3.2 Unexpected Probability Phenomena.	4-4
4.4 Analysis of Simulation Data	4-5
4.4.1 Failure Induced by Measurement Noise Variance Bias or Ramp.	4-5
4.4.2 Failure Induced by Measurement Bias or Ramp. . .	4-6
4.4.3 State Estimation.	4-8
4.4.4 Parameter Estimation.	4-9
4.5 Summary.	4-10
V. Conclusions and Recommendations	5-1
5.1 Conclusions	5-1
5.1.1 MMSOFE.	5-1
5.1.2 Simulations.	5-2
5.2 Recommendations	5-2
Appendix A. New MMSOFE Subroutines	A-1
A.1 MMAE Subroutine	A-1
A.2 OPFILE Subroutine	A-9
Appendix B. Simulation Plots	B-1
B.1 Measurement Noise Bias at $T > 25$	B-2
B.2 Measurement Noise Ramp during $25 < T < 35$	B-11
B.3 Measurement Noise Bias during $25 < T < 35$	B-20
B.4 Measurement Bias to .1 for $T > 10$	B-29

	Page
B.5 Measurement Ramp up to .1 for $10 < T < 20$	B-38
B.6 Measurement Bias to .15 for $T > 10$	B-47
B.7 Measurement Ramp to .15 for $10 < T < 25$	B-56
Bibliography	BIB-1
Vita	VITA-1

List of Figures

Figure	Page
1.1. Types of Failures	1-3
1.2. Multiple GLR Testing	1-8
1.3. One Possible Configuration of Multiple Model Adaptive Estimation . .	1-10
2.1. MMAE-Based Control	2-11
2.2. Multiple Model Adaptive Controller	2-12
2.3. One Possible Distributed Kalman Filter Estimator	2-13
3.1. MSOFE MAIN Program	3-5
3.2. MSOFE SIMRUN Subroutine	3-11
3.3. MSOFE MEASUP Subroutine	3-12
3.4. MMSOFE MAIN Program	3-21
3.5. MMSOFE SIMRUN Subroutine (Part A)	3-23
3.6. MMSOFE SIMRUN Subroutine (Part B)	3-24
3.7. MMSOFE MEASUP Subroutine	3-25
3.8. NEW COMMON BLOCK Variables	3-30
3.9. NEW COMMON BLOCK Variables Descriptions	3-31
3.10. COMMON BLOCK variables	3-32
3.11. MMSOFE.f and USOFE.f Local Variables	3-33
B.1. Case #1 Probabilities	B-3
B.2. Case #1 Range Residuals	B-4
B.3. Case #1 Theta Residuals	B-5
B.4. Case #1 Range	B-6
B.5. Case #1 Range-rate	B-7
B.6. Case #1 Theta	B-8

Figure	Page
B.7. Case #1 Theta-rate	B-9
B.8. Case #1 Parameter	B-10
B.9. Case #2 Probabilities	B-12
B.10. Case #2 Range Residuals	B-13
B.11. Case #2 Theta Residuals	B-14
B.12. Case #2 Range	B-15
B.13. Case #2 Range-rate	B-16
B.14. Case #2 Theta	B-17
B.15. Case #2 Theta-rate	B-18
B.16. Case #2 Parameter	B-19
B.17. Case #3 Probabilities	B-21
B.18. Case #3 Range Residuals	B-22
B.19. Case #3 Theta Residuals	B-23
B.20. Case #3 Range	B-24
B.21. Case #3 Range-rate	B-25
B.22. Case #3 Theta	B-26
B.23. Case #3 Theta-rate	B-27
B.24. Case #3 Parameter	B-28
B.25. Case #4 Probabilities	B-30
B.26. Case #4 Range Residuals	B-31
B.27. Case #4 Theta Residuals	B-32
B.28. Case #4 Range	B-33
B.29. Case #4 Range-rate	B-34
B.30. Case #4 Theta	B-35
B.31. Case #4 Theta-rate	B-36
B.32. Case #4 Parameter	B-37
B.33. Case #5 Probabilities	B-39

Figure	Page
B.34.Case #5 Range Residuals	B-40
B.35.Case #5 Theta Residuals	B-41
B.36.Case #5 Range	B-42
B.37.Case #5 Range-rate	B-43
B.38.Case #5 Theta	B-44
B.39.Case #5 Theta-rate	B-45
B.40.Case #5 Parameter	B-46
B.41.Case #6 Probabilities	B-48
B.42.Case #6 Range Residuals	B-49
B.43.Case #6 Theta Residuals	B-50
B.44.Case #6 Range	B-51
B.45.Case #6 Range-rate	B-52
B.46.Case #6 Theta	B-53
B.47.Case #6 Theta-rate	B-54
B.48.Case #6 Parameter	B-55
B.49.Case #7 Probabilities	B-57
B.50.Case #7 Range Residuals	B-58
B.51.Case #7 Theta Residuals	B-59
B.52.Case #7 Range	B-60
B.53.Case #7 Range-rate	B-61
B.54.Case #7 Theta	B-62
B.55.Case #7 Theta-rate	B-63
B.56.Case #7 Parameter	B-64

List of Tables

Table	Page
1.1. Proposed Orbit Problem Failure Cases	1-15
3.1. MSOFE Input/Output Files	3-2
3.2. MMSOFE Input/Output Files	3-14
4.1. Orbit Problem Failure Cases	4-2
B.1. Orbit Problem Failure Case #1	B-2
B.2. Orbit Problem Failure Case #2	B-11
B.3. Orbit Problem Failure Case #3	B-20
B.4. Orbit Problem Failure Case #4	B-29
B.5. Orbit Problem Failure Case #5	B-38
B.6. Orbit Problem Failure Case #6	B-47
B.7. Orbit Problem Failure Case #7	B-56

List of Symbols

Symbol		Page
$\chi(t_k)$	chi-square random variable	1-9
\mathbf{r}	Residual vector	1-9
$\Lambda(t_j)$	Covariance of the residuals	1-9
T	Simulation time	1-15
R_S	Truth or System (real-world) measurement noise variance . . .	1-16
R_{S0}	Nominal value of R_S	1-16
R_{f0}	Elemental filter nominal range measurement noise	1-16
Z_S	Truth model range measurement	1-16
Z_f	Elemental filter-predicted range measurement	1-16
$BIAS_{R_f}$	Elemental filter scale-factor on R_{f0} for added bias	1-16
$BIAS_{Z_f}$	Elemental filter biases added to Z_{f0}	1-16
$\mathbf{f}[\mathbf{x}(t), t]$	Nonlinear state dynamics vector	2-1
$\mathbf{x}(t)$	State vector	2-1
$\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]$	Nonlinear state dynamics vector	2-2
$\mathbf{u}(t)$	Deterministic control input	2-2
$\mathbf{w}(t)$	Process noise	2-2
$\mathbf{Q}(t)$	Process noise strength	2-2
$\mathbf{z}(t_i)$	Discrete measurement vector	2-2
$\mathbf{v}(t_i)$	Measurement noise	2-2
$\mathbf{R}(t_i)$	Measurement noise strength	2-2
$\mathbf{h}[\mathbf{x}(t_i), t_i]$	Nonlinear observation vector	2-2
$\mathbf{x}_n(t)$	Nominal state vector	2-2
$\delta \mathbf{x}(t)$	Perturbation state vector	2-3
$\mathbf{F}[t; \mathbf{x}_n(t)]$	Linearized state dynamics matrix	2-3
$\widehat{\delta \mathbf{x}}(t)$	Error state estimate vector	2-4

Symbol		Page
$\hat{\mathbf{x}}(t)$	State estimate vector	2-4
$\mathbf{P}(t/t_i)$	State covariance matrix	2-4
$\mathbf{K}(t_i)$	Kalman filter gain	2-5
$p_k(t_i)$	Elemental filter conditional probability	2-7
\mathbf{a}_k	Elemental filter hypothesized parameter values	2-7
$\hat{\mathbf{a}}_k$	MMAE blended parameter estimates	2-8
$\mathbf{E}_{blended}$	MMAE blended state estimation error vector	3-16
$\hat{\mathbf{X}}_{Blended}$	MMAE-blended state estimates	3-16
AFS	Filter/system mapping matrix	3-16

Abstract

Multiple model Kalman Filter (KF) techniques are used extensively for Multiple Model Adaptive Estimation (MMAE), Multiple Model Adaptive Control (MMAC), and Distributed Kalman Filter (DKF) applications to determine Bayesian-blended optimal estimates of states, uncertain parameters, and optimal control signals. Multiple model methods are used for sensor management, Failure Detection and Isolation (FDI), and other Guidance and Control (G&C) applications. A simulation tool called the Multiple Model Simulation for Optimal Filter Evaluation (MMSOFE) has been in this research. MMSOFE is based on the well-benchmarked single Kalman filter tool called Multimode Simulation for Optimal Filter Evaluation (MSOFE). MMSOFE is a highly portable and versatile multiple and single Kalman filter evaluation tool. It is capable of performing simulations with one filter or up to 98 elemental filters in a multiple model adaptive filter structure. It can be adapted easily for other multiple model applications. MMSOFE was applied to failure detection and isolation of measurement jamming- and spoofing-type failures, similar to jamming and spoofing of a Global Positioning System (GPS). A satellite orbit estimation test case was used.

DEVELOPMENT OF A PERFORMANCE EVALUATION
TOOL (MMSOFE) FOR DETECTION OF FAILURES
WITH MULTIPLE MODEL ADAPTIVE ESTIMATION (MMAE)

I. Introduction

Military and civilian aircraft employ a wide variety of Inertial Navigation Systems (INS), Global Positioning System (GPS) receivers, and numerous other navigational aids. While each system can function independently, the greatest advantage is gained through integrating the systems available on a particular aircraft. This integration is frequently accomplished using a Kalman Filter (KF). With improved position and velocity estimates obtained with the KF-based integrated systems, both the purview of applications and system precision increase substantially. A few of these integrated system applications are: precise navigation of preplanned flight trajectories, truly automated landing approach systems, air refueling, formation flying, possible aiding of carrier landing, and superior weapon delivery (be it air-to-air, air-to-surface, surface-to-air, or surface-to-surface).

The design and analysis of KF-based integrated system designs has mostly been accomplished using problem-specific computer simulation software. While this problem-specific simulation/analysis method is functional, generating the model-independent KF-common code is a time consuming, frequently redundant effort and further enhances the opportunity for error – maybe even "hidden" errors which are not readily observable and could go undetected during development. Alternatively, analysis of system designs involving either linear or extended KFs has frequently been accomplished by running computer simulations using a program called Multimode Simulation for Optimal Filter Evaluation (MSOFE) (24). MSOFE provides a software tool for analysis of KF designs involving single KFs. It allows for Monte Carlo simulations of linear or extended Kalman Filters, covariance analysis of linear or linearized filters, or both types of analysis simultaneously.

A primary import of MSOFE is that a complete KF analysis is possible with MSOFE by using its Monte Carlo run simulation capability without conducting flight tests.

However, although MSOFE is an excellent tool for simulating and analyzing designs of single KFs without "reinventing-the-wheel" for each problem, it was not intended to simulate systems which utilize multiple KFs, as required in Multiple Model Adaptive Estimation (MMAE), Multiple Model Adaptive Control (MMAC), or Distributed Kalman Filter (DKF) techniques. MMAE, MMAC, and DKF are discussed in more detail in Section 1.5.2, and the motivation for developing a Multiple Model Simulation for Optimal Filter Evaluation (MMSOFE) is discussed in Sections 1.1 and 1.2.

1.1 Background

MMAE has been used for many applications, including: Failure Detection and Isolation (FDI) of aircraft control surface failures (8) and reconfiguring the flight control system to counter the failure effects in such situations (33), variable stability aircraft flight control (21), forward-looking airborne target tracking system (26), MMAE and moving-bank MMAE for sensing and controlling vibrations in a flexible space structure(7, 11), and adaptive control of a robot arm (29), to name a few. Another prime application for MMAE relates to research being conducted by the Central Inertial Guidance Test Facility (CIGTF), 6585th Test Group, Air Force Systems Command (AFSC), Holloman AFB, NM to determine the vulnerability of the GPS to jamming and spoofing (36).

This latter application was the original impetus for this thesis. MMAE can be utilized to detect and isolate jamming or spoofing of a GPS or the total loss of a GPS pseudorange input to the receiver. Jamming can be simplistically described as bombarding the GPS receiver with broad-band electromagnetic noise. In the case of jamming, there is no effort made to mislead the GPS system into calculating erroneous results, but rather the intent is to prevent reception of the GPS satellite signals entirely. In contradistinction, spoofing is employed with the intentional purpose of providing misleading information to the GPS system and thereby causing the system to calculate erroneous results. Since effective spoofing must be clandestine, the spoofer attempts to mimic the GPS satellite signal while adding slight, misleading modifications to the original signal. Undetected and

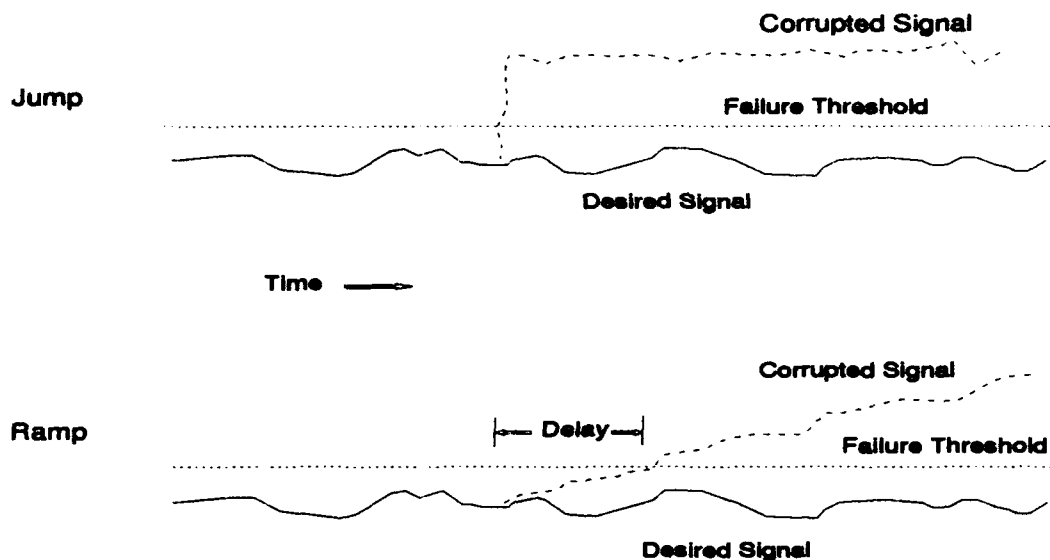


Figure 1.1. Types of Failures

uncountered, this would cause a GPS-dependent-user to be drawn away from his intended destination due to the resulting navigation-related errors. Lastly, another type of failure is merely the total loss of GPS pseudorange measurement data without any hostile intent, but possibly due just to simple antenna shielding or signal attenuation. Serious degradation in performance in the applications mentioned previously would be the final result. While the errors caused by all three types of failure may appear very similar, it is important to identify which of these three types occurred and caused the errors. These types of errors are depicted in Figure 1.1 and generally represent the types of failures simulated to occur in the orbit range measurements for the orbit estimation problem (to be described in detail in Section 3.2) used as the test case for MMSOFE development.

Figure 1.1 depicts two types of failures without specifying the type of desired signal. With regard to this figure, the desired signal could represent a physical quantity such as distance, velocity, acceleration, or impulse or even a value such as noise variance. For the orbit estimation problem, the desired signal represents either the orbit range measurement or the orbit range measurement noise variance. After a specified time, the range measure-

ment signals are corrupted by adding a bias for the jump failure and adding a ramped offset for the ramp failure. In the case of the measurement noise variance, the variance is adjusted by adding a bias or ramp at the time of failure.

In the GPS problem, the same methods could be used for the failure injection (see Section 1.1). The bias and ramp in the measurement noise variance would correspond to the jamming. However, while a jump in the measurement itself could relate to an entire loss in the pseudorange signal, either a jump or ramp in the measurement might be used to simulate simple or intelligent spoofing, respectively.

Additionally, the failure thresholds shown in Figure 1.1 are not readily known *a priori*. The thresholds could be set *a priori* and intelligently if a great deal were known about the trajectory prior to flight and if false failure alarms and missed alarms could be tolerated. A more realistic and practical method would be to use the real-time filter-computed covariances of the residuals to determine the scalar standard deviations, or 1-sigma values (square root of the variances). Some multiple of these 1-sigma values could then be used as threshold values for the residuals to detect failures through residual monitoring or multiple model hypothesis test methods (to be discussed later in more detail).

1.2 Problem Definition

The problem is to develop a simulation and analysis software tool to aid controls designers and system integrators and enable them to concentrate on problems specific to multiple model estimation algorithms, by eliminating the "reinvention" of analysis tool software and thus minimizing the sources of errors resulting from the "reinvention" effort of the software development for each design application. Since MSOFE (4), (24) is a widely known, benchmarked program, it was chosen as a basis upon which to build, rather than building a new independent stand-alone MMSOFE. This will make it much easier for an MSOFE-familiar-user to run MMSOFE without any great learning curve.

Therefore, the primary goal of this thesis is to modify MSOFE to run MMAE algorithms. This modified program will be referred to as Multiple Model Simulation for Optimal Filter Evaluation, or MMSOFE. Due to the adaptive nature of MMAE algorithms,

only Monte Carlo performance evaluations are pursued, and there are no analytically correct equations for the covariance analysis of an MMAE. The Satellite Orbit Estimation problem described in detail by Maybeck (19:pp. 46-48) and in the MSOFE users' manual (24) will be used for the test problem during MMSOFE development. To this basic problem will be added the simulated failures discussed in the previous section to demonstrate MMAE performance attributes. MMSOFE will have application to a wide variety of problems in the Air Force and in industry. MMSOFE will initially be developed to produce optimal blended state estimates and covariances, to calculate parameter estimates, and for failure detection and isolation (FDI). Follow-on goals are to modify MMSOFE further in order to permit MMAE-based control or MMAC, to investigate adapting MMSOFE for systems requiring Distributed Kalman Filters (DKF) as well, and to apply MMSOFE to a larger, more complex problem such as FDI in a GPS/INS integrated system.

1.3 Scope

The failure types considered in this research will be simulated either as a jump or ramp in either the system measurement value or the system measurement noise variance. A jump failure in this context consists of a nearly instantaneous change in the signal offset or noise variance while a ramp failure is a gradually increasing change in the bias measurement offset or noise variance. Unlike many FDI algorithms, it is unnecessary to know and model the exact magnitude of these jumps in advance, since the MMAE algorithm can estimate their magnitude automatically if various hypothesized jump values (hypothesized ramps will not be required) are modeled in the elemental filters. Further, while these type failures may be applicable to many different systems, the primary purpose of this research is to develop the MMSOFE tool itself. The criteria for completion of the research are presented in Section 1.6.

1.4 Assumptions

The assumptions given below provide the scenario for the MMSOFE development and the simulation used in this research. The reader can further investigate their import in the referenced documents.

1. Computer simulations will be accomplished using MSOFE ((4), (24)) and MMSOFE. As described above, MMSOFE is an adaptation of MSOFE to enable MMAE analysis.
2. The MMSOFE user will already be familiar or become familiar with the operation of MSOFE.
3. MSOFE structure will be modified as little as possible in developing MMSOFE.
4. MMSOFE input and output file structure, including MSOFE_IN, CTOM, and DTOM will be essentially identical to MSOFE's for ease of use.
5. MMSOFE will use one MSOFE_IN input file and produce one CTOM and one DTOM output file for each elemental filter and produce one CTOM file for the blended MMAE output and parameter estimation data.
6. MSOFE simulations for elemental filter tuning are performed using covariance analysis and 15-run Monte Carlo analyses with statistical values averaged over the 15 runs. Only Monte Carlo, not covariance, analysis will be used for analysis of the MMAE results.
7. MMSOFE will be structured to allow implementation of MMAE, MMAE-based control, MMAC, or DKF with minimal modifications for follow-on research.
8. The Matrix_x (10) software will be used only for plotting and not for data post-processing; thus the MMSOFE tool will be generally applicable, allowing for full processing and plotting in computer environments that do not include Matrix_x software.

1.5 Literature Review

This section contains a brief review of literature pertaining to several FDI techniques other than MMAE. A review of literature pertaining to KF and MMAE theory and applications is also included, with references mainly from Maybeck (18, 19, 20). The literature review conclusion will discuss the usefulness of MMSOFE with respect to the MMAE-based error detection.

1.5.1 Other Error Detection Methods. One of the simplest and most reliable failure detection techniques is the use of redundant sensing elements for voting. Given a system with triple redundancy, an algorithm can be easily written that will compare the outputs of each element, allowing them to vote on the condition of the other elements. Simply stated, if two elements agree but a third element totally disagrees, the third element is considered faulty and is removed from the system. Without the third element, failures can no longer be isolated by this algorithm. If the two remaining elements disagree, a failure has been detected but not isolated, which requires that both elements be removed from the system to avoid degradation of the system performance. Major disadvantages of direct redundancy lie in the expense, system complexity and logistics of systems with the requisite redundant hardware (6:pp. 1-2).

A more sophisticated approach uses analytic redundancy. By using the physical and dynamic relationships between instruments on an aircraft, it is possible to generate multiple sources of the same information mathematically. These sources are used by the FDI structure through voting techniques. Analytic redundancy avoids the need for excessive identical hardware, by allowing non-identical instruments to provide the sources of signals for the voting scheme. Therefore, only the instruments normally required for specific missions are used without adding duplicate copies of instruments. The surplus of information is also used to provide isolation of the failure without the need for triple hardware redundancy (6:pp. 3-7).

In situations in which direct redundancy is impractical, multiple Generalized Likelihood Ratio (GLR) testing (36, 38) is an alternative which uses a KF to compensate for system failures, resulting in accurate state estimates. Figure 1.2 shows how the sensor, the KF, and the FDI block interact. Three hypotheses are depicted, with H_0 , H_1 , and H_2 representing no failure, jump failure and ramp failure, respectively. The KF is designed based on the H_0 hypothesis, and the additional two matching filters are designed based on the H_1 and H_2 hypotheses (35). When considering hypotheses H_1 and H_2 , the matching filters' design parameters determine what type of failure is being matched without prior knowledge of the jump magnitude or ramp slope, either or both of which are estimated by the GLR algorithm. Each of the matching filters monitor the KF residuals and computes

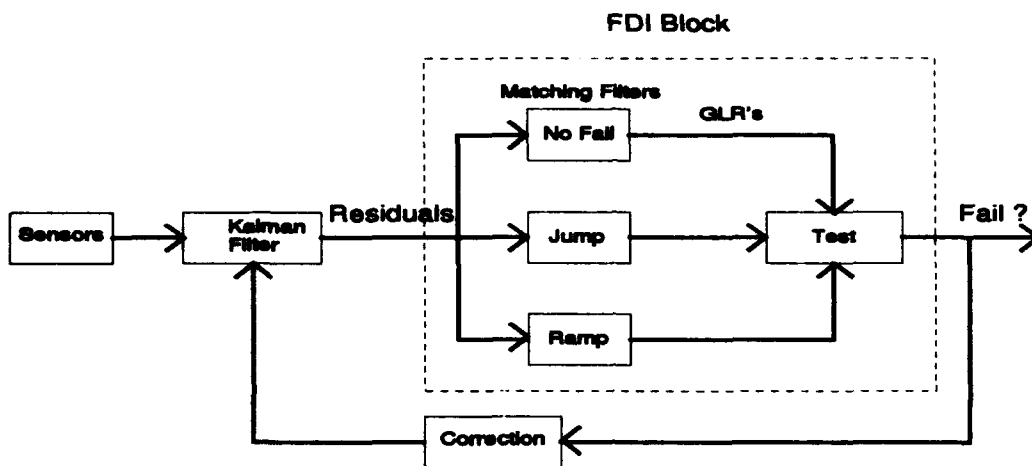


Figure 1.2. Multiple GLR Testing

a maximum likelihood estimate (MLE) of the correctness of the error hypothesis modelled by that matching filter. The maximum likelihood estimates (MLE) determined by each of the hypothesized matching filters are then used in comparison test logic to decide which matching filter is correct. Feedback to the KF can also enable failure adaptation. Frequently, a compensating bias as the feedback signal is all that is needed to continue using the faulty sensor. One advantage of the GLR test over some other FDI algorithms is that prior total knowledge of the failure magnitude is unnecessary. GLRs and MLE's are presented in detail by Willsky and Jones (36, 37, 38).

The simple hypotheses for fail/no-fail conditions may not provide the robustness needed to detect and adapt to certain failures. Detection of ramp failures is particularly difficult when the ramp rate varies significantly. A time delay also exists in detecting ramp failures. This delay occurs because the deceiving signal is slowly moving away from the desired signal and takes longer to cross the failure threshold than with a jump failure. These failure threshold checks might be as simple as monitoring the residuals to see when they exceed some threshold value or they could be more complex methods. The residuals

are particularly susceptible to change when failures are induced as biases or ramps directly in the measurement or in the measurement noise variance.

Considering a filter which hypothesizes no failure, in the case of a true measurement bias/ramp, the residual itself will be consistently offset from its prior value (circa zero hopefully) and in the case of the measurement noise variance bias/ramp failure, the actual covariance of the observed residuals increases when the failure is present. By adding filters designed to match the ramp failure, these longer delays can possibly be avoided. It is also desirable to design the filter based on a specific assumed time of the failure. If the filter were customized to look for a failure at a specific instant in time, then it would have a better chance of accurate detection, but this idea results in a very large bank of filters for long periods of time. Additionally, GLR could estimate the time of onset of a ramp as well as estimate the slope of the ramp. The disadvantage of adding filters is the increase in computations required for multiple filters. A solution to this problem uses a set number of filters over a window of time in which each filter differs slightly, so that one filter will hopefully match the real world. The number of filters remains constant, and the performance of the FDI system is often maintained (38:pp. 606-608). The justification for using sliding windows is discussed by Willsky (38:pp. 604-605).

Another FDI method based on residual monitoring is a chi-square test, which is similar to GLR testing in that it calculates a $\chi(t_k)$ random variable based on the filter residuals, given by

$$\chi(t_k) = \sum_{j=k-N+1}^k \mathbf{r}^T(t_j) \Lambda^{-1}(t_j) \mathbf{r}(t_j) \quad (1.1)$$

with N being the size of a sliding window, \mathbf{r} being the vector of the residuals, and $\Lambda(t_j)$ being the covariance of the residuals. One difference between these two algorithms is that GLR tests are explicitly functions of the assumed system dynamics model and chi-square tests are also but only indirectly. In any case, the chi-square test can only determine if a given hypothesis is correct. A second major difference is that chi-square tests do not try to match the failure and only have one hypothesis, making it a binary test for fail/no-fail. Without the ability to distinguish between types of failures, the chi-square test is not

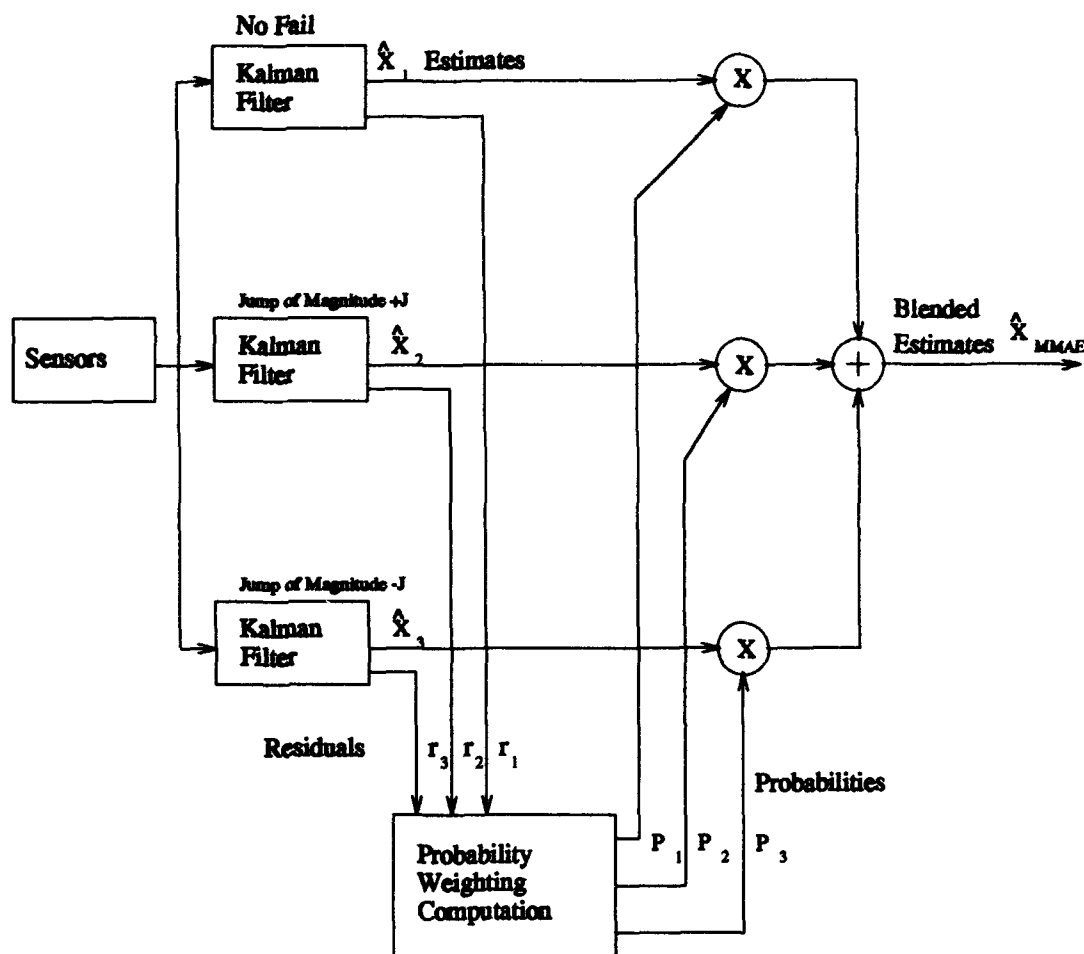


Figure 1.3. One Possible Configuration of Multiple Model Adaptive Estimation

useful for isolation. However, chi-square testing is easy to implement, runs quickly, and can provide the first level of failure detection in a multi-level FDI scheme.

1.5.2 Multiple Model Adaptive Estimation (MMAE). Of primary interest for this research is this final technique, the use of multiple models that represent the dynamics of the system to include various failure conditions. This technique is somewhat analogous to multiple GLR testing in that it explicitly accounts for multiple hypotheses, but it differs in its structure and decision making process and in its capability of improved performance. Unlike the multiple GLR testing which modeled a variety of failures using matching filters, the MMAE models the dynamic nature of the system and its sensors to represent their behavior in the presence of a set of particular assumed failures. Figure 1.3 shows elemental

Kalman filters which are designed to model the various failure conditions. The elemental filter residuals are used to determine which filter best models the aircraft and its sensors at the current time and thus to indicate which hypothesized failure has occurred. A jump or ramp might actually be modeled by several elemental filters, each of which models a different magnitude of bias. It should be noted that ramp failures probably don't require ramp hypotheses, but given bias hypotheses, the MMAE algorithm can switch sequentially from one assumed bias value to another. A probability of adequacy of the model assumed in each elemental filter ranging from zero to one is computed for each elemental filter and is used to weight that filter's state estimates to form a probability-weighted average as an output estimate. A Bayesian form of MMAE filter blends (weighted-averages) the probability-weighted estimates of the respective elemental filters to produce the optimal blended estimates. A Maximum A-Posteriori (MAP) form of MMAE algorithm, in contrast, would select the one elemental filter with the highest probability, and output that single filter's estimate instead. Similarly the parameters which vary between elemental filters can be weighted with the corresponding probabilities to estimate the true value of that parameter. This MMAE blending approach allows for partial sensor failures or combinations of failure types, as in the blending together a non-failure filter and a filter which assumes a jump bias of magnitude J in a specific sensor. This blending could handle an actual jump value between zero and J . However, the discretization of values of J cannot be too coarse or else no filter might be close enough to the real-world bias to generate well-behaved residuals. Probabilities of accuracy ranging from zero to one correspond to filter models that are deemed to be completely inaccurate or perfectly accurate, respectively. A state estimate's probability corresponds directly to the decimal portion of that state estimate which contributes to the blended estimate. MMAE is described in detail by Maybeck (20) and has been used for numerous applications as discussed in Section 1.5.3. An important adaptation of MMAE is that of either MMAE-based control or multiple model adaptive control (MMAC) for system stability and failure correction (related diagrams are shown in Chapter II). MMSOFE should be easy to modify with a user subroutine to accommodate MMAE-based control or MMAC.

1.5.3 MMAE Applications. As mentioned previously in Section 1.1, MMAE has been used for many different applications and most notably at the Air Force Institute of Technology (AFIT) (7, 8, 11, 21, 26, 29, 30, 33), Alphatech Inc. (13), and The Computer Technology Institute (Patras, Greece) (14). MMAE, MMAE-based control and/or MMAC have been the method of choice for a number of applications-oriented-research projects:

1. Detect and identify sensor and aircraft flight control sensor and/or control surface failures (8, 21).
2. Reconfigure an aircraft flight control system when system failures occur (33).
3. Select among different first- and second-order target acceleration models for use in a correlator/forward-looking-infrared tracker for airborne targets (26).
4. Implement an adaptive model-based control for a robotic arm (29).
5. Study target tracking and weapon lead prediction using MMAE (13).
6. Implement a moving-bank subset of multiple models for flexible spacestructure control in the face of uncertain parameters to describe the bending modes (7, 11, 30).

1.5.4 Literature Review Conclusion. The simple error detection methods presented in Section 1.5.1 might be effective for detecting some failures, but may be inadequate for more complex systems such as detecting errors in GPS pseudorange in a GPS-aided inertial navigation system, due to jamming or spoofing. These errors are small in comparison to some of the other GPS errors and are therefore difficult to detect. While such errors would degrade the GPS performance, the cause of the failures would still be undetectable with such a simple error detection method (5).

Direct redundancy techniques using multiple GPS receivers or extra satellites to provide redundancy are expensive and normally impractical, but not impossible. However, analytical redundancy is inherent to integrated navigation systems in that the sensors (i.e., GPS, INS, barometric altimeter, and etc.) can be used to provide multiple sources of the same information.

Multiple GLR testing and MMAE are prime candidates for an FDI system. Both of these techniques possess the FDI versatility needed for complex systems, but their major

difference is in the type of multiple filters required by each and the accuracy of the state estimates. Multiple GLR testing uses a bank of matching filters that are somewhat less complicated than the bank of KFs used in MMAE. Quicker response might be anticipated in MMAE because the elemental filters are running in parallel at all times based on competing hypotheses versus only one filter. Vasquez (36) considered both techniques as candidates for his FDI algorithm and chose to use the GLR tests and chi-square tests. He used the GLR algorithm only for failure isolation, but MMAE should be capable of simultaneous failure detection, failure isolation, and generation of blended estimates which are superior to the estimates from any of the MMAE elemental filters or those resulting from the GLR method. Performance evaluation of those other approaches can be, and has been, established with the standard MSOFE (or similar) analysis tool. Therefore, the MMSOFE-MMAE tool developed in this thesis will make it much easier to compare MMAE FDI results with results from any of the aforementioned methods.

1.6 Methodology

The main steps of this research are explained in Sections 1.6.1 through 1.6.5. Steps one and two will satisfy the primary goal of the thesis, while step three is aimed at the follow-on goal included as a recommendation for future research.

1.6.1 MMAE Implementation Options. There were several methods considered for implementing MMAE using MSOFE.

1. Generating new general-use MMAE-specific software would always be an option, either with or without using segments of MSOFE. This would mean the generation of new software which would not be familiar to MSOFE users and therefore require a greater learning curve for a new user than would be required with an MSOFE-similar approach like in 4 below.
2. One obvious method is to run MSOFE serially, once for each elemental filter and then use post-processing to calculate the probabilities, blended state estimates, parameter estimates and other desired values. For simple MMAE, this is a viable option, although the post-processing algorithms would probably be re-invented for each ap-

plication because the particular software package used for post-processing might not be available or familiar to the user. Moreover, true feedback for MMAE-based control would still not be possible.

3. Another method using MSOFE would be to add a set of states for each elemental filter to the "filter" model used in MSOFE, thus forming one large filter. This one filter would be equivalent to the bank of elemental filters, and the blending computation would have to be augmented to that "single filter" algorithm. With this implementation, the number of filter states in the MSOFE model would equal the number of states required by the problem multiplied by the number of elemental filters. This approach would be possible but the total number of states would grow rapidly for more complex problems. Further, the MMAE probability computing and estimate blending algorithms would need to be programmed separately or possibly as even more additional states. Therefore, a four-state MMAE problem with four elemental filters might require twenty states. MMAE-based control could be possible but implementation would be complex.
4. Last is a method which would truly implement MMAE. It would calculate a propagation and update cycle for each elemental filter, followed by the MMAE-specific algorithms, before any filter would begin another propagation cycle. This method was chosen, because MSOFE could be modified to preserve its features and because this method allows any multiple model application, be it MMAE, MMAE-based control, or DKF to be run with minimal user modifications.

1.6.2 Software Development. As mentioned in the literature review, multiple GLR testing has an advantage over MMAE of needing only one KF, while MMAE requires a bank of such filters running in parallel. Contrarily, MMAE has the advantages of parameter estimation, FDI, improved state estimates and control in the MMAE-based control form. Moreover, the very existence of a parallel bank of filters, each based on a different hypothesis and each producing residuals associated with a particular hypothesis, inherently aids the hypothesis testing process in the MMAE. Many previous AFIT Kalman filter theses utilized single-filter MSOFE runs on the VAX, but with MMSOFE running multiple filters in

parallel, MMAE on the VAX would take entirely too long. Therefore, the satellite orbit problem was used with MSOFE and MMSOFE on the SPARC workstations. MSOFE was modified into MMSOFE to run MMAE, and MMSOFE was verified using the satellite orbit problem as presented by Maybeck (19:pp. 46-48) and in the MSOFE users' manual (24) as a test case. Further, the data output from the each elemental filter was compared with the output for the corresponding filter run with MSOFE. This permitted a bit-by-bit comparison of the data produced by MMSOFE with that of MSOFE, demonstrating that the implementation and performance assessment of the elemental filters within the MMAE were error-free.

1.6.3 Preliminary Studies. The satellite orbit problem used as the test case for this research has range and time measured in generic radius-units and time-units. This study covers both ramp and jump failures in the range measurement and range measurement noise variance at different magnitudes and rates, as shown in Table 1.1. The primary goal of this satellite orbit problem will be software validation with the intrinsic goals of accomplishing FDI in the range estimate of the satellite. No effort will be made to adapt to any failures.

Table 1.1. Proposed Orbit Problem Failure Cases

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
1	Step in Measurement Noise	$R_S = R_{S0} + 6R_{S0}$	$T > 25$	$R_{f0}(1 + BIAS_{R_f})$
2	Ramp in Measurement Noise	$R_S = R_{S0} + \frac{6R_{S0}(T-25)}{(15)}$ $R_S = R_{S0} + 6R_{S0}$	$25 < T < 35$ $T > 35$	$R_{f0}(1 + BIAS_{R_f})$
3	Step/Return in Measurement Noise	$R_S = R_{S0}$ $R_S = R_{S0} + 6R_{S0}$	$T < 25, T > 35$ $25 < T < 35$	$R_{f0}(1 + BIAS_{R_f})$
4	Step in Measurement	$Z_S = Z_{S0} + .1Z_{S0}$	$T > 10$	$Z_f + BIAS_{Z_f}$
5	Ramp in Measurement	$Z_S = Z_{S0} + \frac{.1Z_{S0}(T-10)}{(10)}$ $Z_S = Z_{S0} + .1Z_{S0}$	$10 < T < 20$ $T > 20$	$Z_f + BIAS_{Z_f}$

To explain the entries in Table 1.1 further:

T = Current simulation time.

R_S = Truth model (real-world or "System") range measurement noise variance.

R_{S0} = Nominal value of R_S .

R_{f0} = Elemental filter nominal range measurement noise variance.

Z_S = Truth model range measurement.

Z_f = Elemental filter-predicted range measurement.

$BIAS_{R_f}$ = Elemental filter scale-factor on R_{f0} for bias added to R_{f0} ($BIAS_{R_f} = \{6, 3, 0, -.5, -.75\}$).

$BIAS_{Z_f}$ = Elemental filter biases added to Z_{f0}
($BIAS_{Z_f} = \{.1, .05, 0, -.05, -.1\}$).

The onset and relaxation times for the failures were chosen so there would be enough time both before and after the failures for the elemental filters' residuals to stabilize at some "steady state" value, thus resulting in MMAE-calculated probabilities shifting from baseline to the appropriate elemental filters after induction of the failure. The magnitude of the failures in the truth models were chosen to be large enough to make it possible for good discrete failure detection and isolation among the elemental filters, while still not being so large as to be totally unrealistic. The magnitudes chosen for $BIAS_{R_f}$ and $BIAS_{Z_f}$ were chosen to make the failure detection obvious, with the largest values in each case matching the magnitude of the truth model failure bias or, in the case of ramps, matching the final (sustained) magnitude. The other magnitudes of $BIAS_{R_f}$ were chosen such that the resulting noise variance would be $\frac{1}{2}$ of the prior value. Once again for $BIAS_{Z_f}$, the largest magnitude matches that in the system model, with the second value being $\frac{1}{2}$ the first value, followed by the baseline filter which hypothesizes no failure. The last two are just the negative (mirror image) of the first two.

1.6.4 Software Validation Tests. Because development of the MMAE software tool, MMSOFE, is the purpose of this thesis, software validation is paramount to this research, particularly since MMSOFE will be an entirely new adaptation of MSOFE which is not without risk. The steps in testing the MMSOFE software include:

1. Modify the orbit MSOFE SPARC code currently available for the chosen simulated errors (see Table 1.1).
2. Conduct the MSOFE runs shown in Table 1.1 and evaluate the results.
3. Tune the filters which represent each failure type via MSOFE covariance analysis runs.
4. Generate the new SPARC MMSOFE orbit code by modifying the SPARC MSOFE orbit code for the MMAE algorithm.
5. Determine the filter models and elemental filters required in MMAE for the failure types shown in Table 1.1.
6. Partially validate the MMSOFE software by using multiple, but identical elemental filters with the MMAE probability and blending computations disabled. Compare the output with that obtained from MSOFE.
7. Validate the MMSOFE software by using multiple, but distinct elemental filters and compare the elemental filter results to data obtained from matching MSOFE runs.
8. Generate MMSOFE to employ Bayesian-blending for the optimal state and parameter estimation.
9. Run MMSOFE with no Hohmann orbit transfer for the first four failure types shown in Table 1.1 and compare the blended state estimates to the output of the elemental filters.
10. Possibly run MMSOFE with Hohmann orbit transfer for the first four failure types shown in Table 1.1 and compare the blended state estimates to the output of the elemental filters.

1.6.5 Follow-on Goals. As discussed above, one follow-on goal is to adapt MMSOFE to provide the optimal corrective feedback and thus make the MMAE system into a truly adaptive MMAE-based control system or MMAC. Another follow-on goal is to investigate adapting MMSOFE for Distributed Kalman Filter (DKF) applications as well. Further, MMSOFE could be used merely to provide multiple tuning runs by using elemental filter with identical models but different tuning values versus making multiple MSOFE

tuning runs with the different tuning values. An added benefit to using MMSOFE in this manner would be the parameter estimation to aid in determining the correct tuning values.

Lastly, a follow-on application-oriented goal is to use MMSOFE on the Navigation Reference System (NRS) FDI problem investigated by Vasquez (36) who used GLR and chi-square techniques. Adaptive corrective feedback could improve the state estimates and FDI capabilities of a MMAE-blended NRS system.

1.6.6 Stopping Criteria. The primary stopping criteria correlate to the steps in Section 1.6.4. The stopping criteria for software development and validation are as follows:

1. MSOFE orbit estimation problem tuning runs with the errors shown in Table 1.1 will be complete when the true error standard deviations computed in covariance analysis roughly match the filter-predicted values.
2. MMSOFE algorithm will be validated when the output for each elemental filter exactly matches that produced by running the identical model with MSOFE, first when all elemental filters match and then when they differ.
3. The MMAE blending algorithm will be complete when the blended state estimates are reasonable in comparison to the state estimates produced by the elemental filters.

1.7 Overview of Thesis

Chapter II presents the detailed theory used in the research. KF theory is discussed, including time-discretization of the dynamics for sampled data KF implementation and with special attention on MMAE, MMAE-based control and MMAC, and some attention to DKF. The basics of the FDI algorithms are discussed, including the equations implemented for the MMAE.

Chapter III contains the detailed description of the MMSOFE adaptation of MSOFE, to include the theoretical basis, the software development and detailed modifications to MSOFE. The basics of the orbit estimation problem are discussed, as well as the results from the MMSOFE preliminary verification simulations.

Chapter IV describes results of the MMSOFE orbit problem simulations. Analysis of the elemental filter performance, the blended state estimates, the probabilities and parameter estimation, and the FDI results will also be discussed.

Chapter V summarizes the research and presents conclusions and recommendations with regard to using MMSOFE for other multiple model applications.

II. Kalman Filtering, MMAE, and Failure Detection

2.1 Overview

This chapter presents the fundamental theory of Kalman filtering, both when used as a single filter and as employed in a bank of elemental filters in the MMAE context. Both MSOFE and MMSOFE can run linear as well as extended Kalman filter simulation. Because the satellite orbit problem used for MMSOFE development in this thesis uses an extended Kalman filter (EKF) algorithm, a suitable discretization method of the system dynamics must be used. The method of discretizing the filter equations for continuous-time dynamics will be described. MMAE-related FDI methods and MMAE blended estimation will be discussed. Stochastic adaptive controllers in the form of MMAE-based controllers and Multiple Model Adaptive Controllers, and Distributed Kalman Filtering will also be discussed with respect to MMSOFE. Finally, filter tuning and selection of lower bounds for MMAE-computed hypothesis conditional probabilities will be discussed briefly.

2.2 The Extended Kalman Filter

Because of the dynamic nature of the chosen problem of interest and for the sake of more general applicability, an Extended Kalman Filter (EKF) was chosen to provide state estimates. The EKF allows for nonlinear, time-varying dynamics and/or measurement equations. In simple linearized Kalman filtering (LKF), these equations are linearized through approximation techniques about a *fixed* nominal trajectory to form the LKF. The LKF is the conceptual basis for the EKF, but the EKF is continually *relinearized* about the most recent state estimate rather than about a fixed nominal trajectory.

2.2.1 The Sampled Data Kalman Filter. Let the system model be expressed as a state equation of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), t] + \mathbf{G}(t)\mathbf{w}(t) \quad (2.1)$$

where the state dynamics vector $\mathbf{f}[\mathbf{x}(t), t]$ is a nonlinear function of the state vector $\mathbf{x}(t)$ and time. It is noteworthy to point out that this discussion of LKF and EKF equations

correlates very closely with the discussion in Section 9.5 of Maybeck (19). However, there is one simplification in the state dynamics vector, $\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]$ Maybeck's Equation (9-34b), where $\mathbf{u}(t)$ represents the deterministic control inputs. In general extended Kalman filtering and with MSOFE and MMSOFE, the $\mathbf{u}(t)$ inputs are non-zero and non-constant. For the MMSOFE development simulations, however, it is assumed that $\mathbf{u}(t) = \mathbf{0}$, which corresponds to no deterministic control inputs. With that aside, let the process noise, $\mathbf{w}(t)$ be a white Gaussian noise with mean:

$$E \{ \mathbf{w}(t) \} = \mathbf{0} \quad (2.2)$$

and noise strength $\mathbf{Q}(t)$ defined by:

$$E \{ \mathbf{w}(t) \mathbf{w}^T(t + \tau) \} = \mathbf{Q}(t) \delta(\tau) \quad (2.3)$$

The discrete-time measurements, $\mathbf{z}(t_i)$ are modeled as a nonlinear vector of functions of the state vector and time, $\mathbf{h}[\mathbf{x}(t_i), t_i]$ plus additive white Gaussian noise:

$$\mathbf{z}(t_i) = \mathbf{h}[\mathbf{x}(t_i), t_i] + \mathbf{v}(t_i) \quad (2.4)$$

where $\mathbf{v}(t_i)$ is a zero-mean discrete-time white Gaussian noise of covariance $\mathbf{R}(t_i)$ defined by:

$$E \{ \mathbf{v}(t_i) \mathbf{v}^T(t_j) \} = \begin{cases} \mathbf{R}(t_i) & \text{for } t_i = t_j \\ \mathbf{0} & \text{for } t_i \neq t_j \end{cases} \quad (2.5)$$

and $\mathbf{h}[\mathbf{x}(t_i), t_i]$ is the nonlinear observation (measurement) vector. The LKF is based on *perturbation* states about a fixed nominal state trajectory $\mathbf{x}_n(t)$ satisfying the initial condition, $\mathbf{x}_n(t_0) = \mathbf{x}_{n_0}$ (some value, assumed to be known), and

$$\dot{\mathbf{x}}_n(t) = \mathbf{f}[\mathbf{x}_n(t), t] \quad (2.6)$$

where $\mathbf{f}[\cdot, \cdot]$ is shown in Equation (2.1). The corresponding nominal discrete-time measurements for the nominal trajectory are based on the nominal states and time, but without

any additive noise:

$$\mathbf{z}_n(t_i) = \mathbf{h}[\mathbf{x}_n(t_i), t_i] \quad (2.7)$$

The perturbation states are found by subtracting the nominal states in Equation (2.6) from the original states in Equation (2.1):

$$[\dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_n(t)] = \mathbf{f}[\mathbf{x}(t), t] - \mathbf{f}[\mathbf{x}_n(t), t] + \mathbf{G}(t)\mathbf{w}(t) \quad (2.8)$$

Equation (2.8) is approximated to first order through a truncated Taylor series expansion:

$$\dot{\delta\mathbf{x}}(t) = \mathbf{F}[t; \mathbf{x}_n(t)] \delta\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \quad (2.9)$$

where $\delta\mathbf{x}(t)$ are the perturbation states. The definitions for $\mathbf{G}(t)$ and $\mathbf{w}(t)$ are unchanged and the new linearized dynamics matrix $\mathbf{F}[t; \mathbf{x}_n(t)]$ is found by taking partial derivatives of $\mathbf{f}[\mathbf{x}(t), t]$ with respect to $\mathbf{x}(t)$ and evaluated along the nominal values for the trajectory $\mathbf{x}_n(t)$:

$$\mathbf{F}[t; \mathbf{x}_n(t)] = \left. \frac{\partial \mathbf{f}[\mathbf{x}(t), t]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_n(t)} \quad (2.10)$$

The discrete-time perturbation measurements are similarly approximated to first order from the measurement difference equation:

$$[\mathbf{z}(t) - \mathbf{z}_n(t)] = \mathbf{h}[\mathbf{x}(t_i), t_i] - \mathbf{h}[\mathbf{x}_n(t_i), t_i] + \mathbf{v}(t_i) \quad (2.11)$$

yielding the perturbed form:

$$\delta\mathbf{z}(t_i) = \mathbf{H}[t_i; \mathbf{x}(t_i)] \delta\mathbf{x}_n(t) + \mathbf{v}(t_i) \quad (2.12)$$

The same partial derivative methods used to derive the linearized state dynamics matrix $\mathbf{F}[t; \mathbf{x}_n(t)]$ are used again to derive the linearized observation matrix:

$$\mathbf{H}[t_i; \mathbf{x}_n(t_i)] = \left. \frac{\partial \mathbf{h}[\mathbf{x}(t_i), t_i]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_n(t_i)} \quad (2.13)$$

Because the LKF generates error state estimates, $\widehat{\delta \mathbf{x}}(t)$, they must be added to the nominal states to provide whole states estimates $\hat{\mathbf{x}}(t)$ in the form:

$$\hat{\mathbf{x}}(t) = \mathbf{x}_n(t) + \widehat{\delta \mathbf{x}}(t) \quad (2.14)$$

However, if the nominal and the "true" trajectories differ too greatly, a linearized Kalman filter can't really be used because the condition for negligibility of higher order terms in the Taylor series expansions has been violated. In such situations, an extended Kalman filter can be implemented by linearizing about the current state estimate $\hat{\mathbf{x}}$ rather than about the nominal trajectory \mathbf{x}_n . The following sampled data extended Kalman filter equations use the notation t/t_i to represent the value of a given variable at time t , conditioned on the measurements taken through the time t_i . Also, t_i^- represents the value after propagation but prior to the measurement update at sample time t_i , and t_i^+ corresponds to the value after the measurement update. The state estimates $\hat{\mathbf{x}}$ and covariance values $\mathbf{P}(t/t_i)$ are propagated from t_i to t_{i+1} by solving the following differential equations:

$$\dot{\hat{\mathbf{x}}}(t/t_i) = \mathbf{f}[\hat{\mathbf{x}}(t/t_i), t] \quad (2.15)$$

$$\dot{\mathbf{P}}(t/t_i) = \mathbf{F}[t; \hat{\mathbf{x}}(t/t_i)]\mathbf{P}(t/t_i) + \mathbf{P}(t/t_i)\mathbf{F}^T[t; \hat{\mathbf{x}}(t/t_i)] + \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^T(t) \quad (2.16)$$

where

$$\mathbf{F}[t; \hat{\mathbf{x}}(t/t_i)] = \left. \frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(t/t_i)} \quad (2.17)$$

and initial conditions are given by:

$$\hat{\mathbf{x}}(t_i/t_i) = \hat{\mathbf{x}}(t_i^+) \quad (2.18)$$

$$\mathbf{P}(t_i/t_i) = \mathbf{P}(t_i^+) \quad (2.19)$$

The discrete-time measurements are processed in the EKF through update equations:

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T[t_i; \hat{\mathbf{x}}(t_i^-)] \left\{ \mathbf{H}[t_i; \hat{\mathbf{x}}(t_i^-)]\mathbf{P}(t_i^-)\mathbf{H}^T[t_i; \hat{\mathbf{x}}(t_i^-)] + \mathbf{R}(t_i) \right\}^{-1} \quad (2.20)$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i) \{ \mathbf{z}_i - \mathbf{h}[\hat{\mathbf{x}}(t_i^-), t_i] \} \quad (2.21)$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i) \mathbf{H}[t_i; \hat{\mathbf{x}}(t_i^-)] \mathbf{P}(t_i^-) \quad (2.22)$$

where

$$\mathbf{H}[t_i; \hat{\mathbf{x}}(t_i^-)] = \left. \frac{\partial \mathbf{h}[\mathbf{x}(t_i), t_i]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(t_i^-)} \quad (2.23)$$

and $\mathbf{K}(t_i)$ is the discrete-time Kalman filter gain.

It is noteworthy that the covariance updates, Equation (2.22), are actually implemented in MSOFE using $\mathbf{U} - \mathbf{D}$ covariance factorization (24). This covariance update form is shown by:

$$\mathbf{P}(t_i^-) = \mathbf{U}(t_i^-) \mathbf{D}(t_i^-) \mathbf{U}^T(t_i^-) \quad (2.24)$$

$$\mathbf{P}(t_i^+) = \mathbf{U}(t_i^+) \mathbf{D}(t_i^+) \mathbf{U}^T(t_i^+) \quad (2.25)$$

in which the \mathbf{U} is a unitary (diagonal terms, which are the eigenvalues, equal one), upper triangular matrix and the \mathbf{D} matrix is diagonal. The numerical advantages inherent to $\mathbf{U} - \mathbf{D}$ covariance factorization are: guaranteed nonnegativity of the covariance matrix, numerical accuracy, and numerical stability (18). Because MMSOFE is an MSOFE derivative, the same $\mathbf{U} - \mathbf{D}$ covariance factorization advantages are preserved in MMSOFE.

2.3 Multiple Model Adaptive Estimation

MMAE is capable of failure detection and isolation, optimal state estimation and parameter estimation. MMAE is composed of multiple Kalman filters running in parallel (simultaneously), as discussed in Section 1.5.2. Each elemental filter models one of the possible configurations of the systems. These configurations may be intentional, such as elemental filters based upon different possible system parameter values, or the configurations may represent some type of failure, such as those caused by GPS jamming or spoofing. If these failures are of unknown magnitudes, the MMAE must estimate both the states and the failure magnitude by using elemental filters with different assumed failure magnitudes.

This is only slightly different from estimation of an unknown system parameter. Given jumps or ramps in the measurement signals or in the measurement noise variances as failure types, the elemental filters will use different assumed bias values, and the magnitude of the true failure can be estimated by merely summing the probability-weighted assumed values that were used in the elemental filters. There is no need to hypothesize elemental filters based on a ramp failure of an assumed slope and assumed time of onset; as the true ramp occurs, the elemental filter with the bias value that most closely matches the current ramp value ought to achieve the highest probability weighting. This magnitude estimation develops naturally after calculating the conditional probabilities for each elemental filter. These conditional probabilities, also sometimes referred to as hypothesis conditional probabilities, will be used as weights for blending the state estimates from the corresponding elemental filters to determine the blended MMAE estimates. These conditional probabilities are also used to estimate the uncertain parameter value, which in the case of FDI is the true magnitude of the jump or ramp.

Out of all of the possible filters existing in the entire parameter space, the approach in this research will be to use several elemental Kalman filters which differ only by the assumed measurement noise variance or measurement signal bias magnitude. Measurement signal biases would model measurement jumps and/or ramps as might result from intentional spoofing by an adversary, and increased measurement noise variance would model jamming-type real world measurement noise. Nominally, for the case of measurement signal biases, two filters with different positive bias magnitudes, a matched set with negative magnitudes, and one filter with bias magnitude = 0 can be assumed (i.e., $K = 5$, where K is the number of elemental Kalman filters). Using varying variances of measurement noise instead of biases in another set of elemental filters could similarly model jamming noise. Thus, the probabilities determined for each of these filters will correspond to their relative closeness to the truth model (real world) characteristics. The conditional probabilities corresponding to a good filter model (small residuals) will be greater than the conditional probabilities corresponding to a poor filter model (large residuals). Thus the elemental filter state estimates are weighted in proportion to the probability that the specific elemental filter hypothesis about jamming/spoofing status is correct (19). The conditional

probabilities $p_k(t_i)$ (see Figure 1.3) for $k = 1, 2, \dots, K$ are determined by:

$$p_k(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_k, \mathbf{Z}_{i-1})p_k(t_{i-1})}{\sum_{j=1}^K f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_j, \mathbf{Z}_{i-1})p_j(t_{i-1})} \quad (2.26)$$

where, for this development, the \mathbf{a}_k parameters may represent measurement bias magnitudes and/or measurement noise strengths.

The numerator of this equation is the product of two terms. $p_k(t_{i-1})$ is merely the most recent prior value of the conditional probability for the k^{th} elemental filter; Equation (2.26) is an *iteration*. All K numerator terms for the K elemental filters at t_{i-1} must, of course, be available before the denominator and consequently the k^{th} probability at the current time, t_i , can be calculated. The first numerator term is the probability density function of the current measurements conditioned on both the assumed parameter values and observed past measurements. This probability density function is computed by:

$$\begin{aligned} f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_k, \mathbf{Z}_{i-1}) &= \frac{1}{(2\pi)^{\frac{m}{2}} |\Lambda_k(t_i)|^{\frac{1}{2}}} \exp \{ \cdot \} \\ \{ \cdot \} &= \left\{ -\frac{1}{2} \mathbf{r}_k^T(t_i) \Lambda_k^{-1}(t_i) \mathbf{r}_k(t_i) \right\} \end{aligned} \quad (2.27)$$

where m is the measurement vector dimension, and the filter residual is given by:

$$\mathbf{r}_k(t_i) = [\mathbf{z}(t_i) - \mathbf{H}_k(t_i) \hat{\mathbf{x}}_k(t_i^-)] \quad (2.28)$$

and where the residual covariance is computed by the k^{th} elemental filter as:

$$\Lambda_k(t_i) = [\mathbf{H}_k(t_i) \mathbf{P}_k(t_i^-) \mathbf{H}_k^T(t_i) + \mathbf{R}_k(t_i)] \quad (2.29)$$

The k^{th} filter residual, $\mathbf{r}_k(t_i)$, is dependent upon the current measurement, $\mathbf{z}(t_i)$, the measurement matrix $\mathbf{H}_k(t_i)$ and the state estimate, $\hat{\mathbf{x}}_k(t_i^-)$, before the i^{th} measurement. The k^{th} elemental filter's state estimation error covariance matrix before the i^{th} measurement, $\mathbf{P}_k(t_i^-)$, the measurement matrix and the observation noise covariance matrix, $\mathbf{R}_k(t_i)$, are used to construct $\Lambda_k(t_i)$ (19:131). The Kalman filter equations and notation were discussed in Section 2.2.1.

If the residuals in filter k are in consonance with the filter-computed $\Lambda_k(t_i)$, the quadratic in Equation (2.27) is about equal to $-\frac{m}{2}$, whereas if the residuals are much bigger than anticipated due to the wrong parameter hypothesis, then the quadratic is a much larger negative number, so p_k decreases substantially. The denominator of Equation (2.26) represents the sum of the numerator terms from each elemental filter computed at time, t_i . This scales the numerator to ensure that the sum of the p_k 's is always one. However, a difficulty arises if the conditional probability of a state estimate were to become zero. In such case, the conditional probability, p_k , becomes "locked" at a zero value due to the iterative form of Equation (2.27). Real world changes will not unlock p_k from zero even if the elemental filter is producing good state estimates. Therefore, p_k , in Equation (2.26) would remain zero for all time because $p_k(t_{i-1})$, the most recent value of the conditional probability, multiplies the other terms in Equation (2.26). Therefore, a lower bound (threshold) on p_k should be set to preclude such a zero lock-in condition.

The state estimates, \hat{x}_k , are then scaled (multiplied) by the appropriate weighting factor, p_k , for each elemental filter and these weighted state estimates are summed for all K filters. Summing these products results in the blended or adaptive state estimates:

$$\hat{x}(t_i^+) = \sum_{k=1}^K \hat{x}_k(t_i^+) \cdot p_k(t_i) \quad (2.30)$$

The covariance, $P(t_i^+)$, of the blended solution is estimated by:

$$P(t_i^+) = \sum_{k=1}^K p_k(t_i) \{ P_k(t_i^+) + [\hat{x}_k(t_i^+) - \hat{x}(t_i^+)] [\hat{x}_k(t_i^+) - \hat{x}(t_i^+)]^T \} \quad (2.31)$$

The parameter estimates, \hat{a}_k are calculated in the same manner as the state estimates, merely by scaling the assumed value of the parameter from each elemental filter by the corresponding weighting factor, p_k , and summing these weighted estimates for all K filters according to the relationship:

$$\hat{a}(t_i) = \sum_{k=1}^K a_k \cdot p_k(t_i) \quad (2.32)$$

This Bayesian form of adaptive estimation is depicted in Figure 1.3. Other approaches are possible (19) but will not be considered in this research.

2.4 Failure Detection

This section presents the theory behind MMAE for the purpose of failure detection. Given a single KF as developed in Section 2.2, GLR and chi-square algorithms could be used to observe changes in the residuals. For significant changes in the residuals, the GLR or chi-square values would exceed some designated threshold value. If the GLR or chi-square values exceed that threshold, a failure is indicated. GLR and chi-square methods are mentioned only for comparison purposes and because they were the methods used by Vasquez (36) and several others for their research. After completion of this Section 2.4, only FDI using MMAE will be discussed in any detail in the remainder of this thesis.

2.4.1 FDI via MMAE. This discussion is presented to supplement that which was presented in Section 2.3. A simplifying fact is that failure detection and isolation using MMAE becomes virtually automatic once the MMAE has been developed, as shown in Section 2.3. Failure detection and isolation are accomplished merely by observing the conditional probabilities, p_k , for the elemental filters. A failure has been detected and identified when the p_k for the baseline (i.e., no-fail) elemental model decreases and p_k for another elemental model increases. The elemental filter with the highest p_k contains the model which most closely matches the truth model (real world). If one of the elemental filters contained an assumed parameter value which exactly matched the actual failure magnitude, the failure modeled in that elemental filter has therefore been isolated as the failure which occurred. However, if none of the elemental filters exactly model the true error, which would be the most common occurrence, the parameter estimation ability could then be used to identify the failure magnitude which lies between discrete values of the discretized parameter space. This is because the parameter estimation capability would use the K probabilities to weight the assumed parameter values and estimate a parameter value which would approximate the true failure magnitude.

2.4.2 Threshold Selection and Filter Tuning. Revisiting GLR and chi-square FDI systems, the thresholds must be selected such that they are low enough to permit failure identification and to minimize delays in detecting failures. Also, they must be low enough to prevent missed alarms caused by GLR or χ values dropping below the threshold while

an actual failure is still present. Finally, the thresholds must be high enough to prevent false alarms caused by variations in the GLR and χ values. These variations may result from aircraft maneuvers or unpredictable changes in the random measurement noise or from other unmodelled effects. If the filter is tuned sufficiently for both good tracking and enhanced failure detection, such variations due to aircraft maneuvers or unpredictable changes in the random measurement noise define a relatively low bound set by the noise magnitude. Analogously, assuming adequate tuning of the elemental filters, thresholds for FDI with MMAE are accomplished by observing the elemental filter probabilities ranging from 0 to 1 such that failures are detected and identified by the elemental filter probabilities. Lower bound on the probabilities must be set appropriately, high enough to prevent zero lock-in but low enough still to allow meaningful probability calculations. If the lower bound is too high, probabilities below that value are impossible to attain, and the adaptation is incapacitated because too little of the total probability is still available to be moved around among the elemental filters.

When tuning a Kalman filter, a major tradeoff must be made to meet the goals for state estimation and failure detection. Adjusting the process noise and measurement noise values to enhance FDI capabilities may degrade state estimation and vice versa. This can be seen in Equation (2.29) by realizing that reductions in \mathbf{R} will cause reductions in $\mathbf{\Lambda}$. The consequence is that reductions in \mathbf{R} will cause an increase in the ratio of \mathbf{Q} eigenvalues to \mathbf{R} eigenvalues, resulting in a conservatively tuned filter which may yield performance degraded from the performance achievable when the filter is tuned specifically for best state estimation performance. Decreasing \mathbf{R} and increasing \mathbf{Q} could yield the same tuning result, though, since the steady state gain, \mathbf{K} , depends on the ratio of \mathbf{Q} and \mathbf{R} eigenvalues. Observation of the state estimates, measurement residuals and the residual covariance $\mathbf{\Lambda}(t_i)$ will indicate a point of diminishing returns using this technique. The elemental filters can be tuned either via multi-run MSOFE simulations or tuned using MMSOFE with different tuning values used in the different elemental filters.

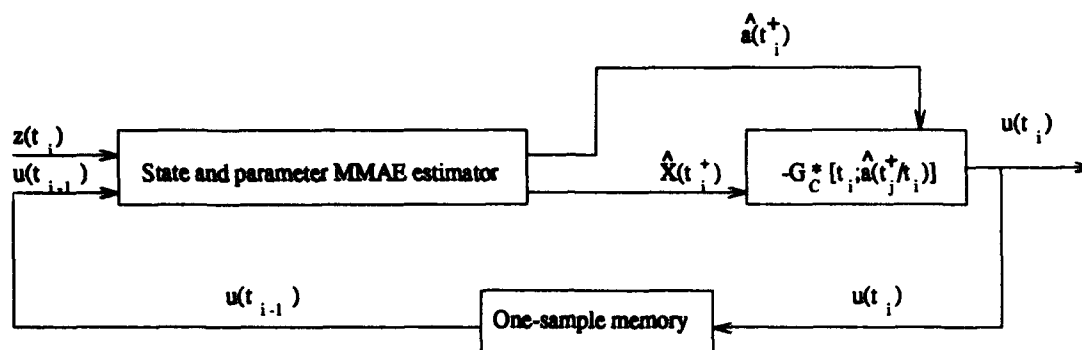


Figure 2.1. MMAE-Based Control

2.5 Stochastic Adaptive Control

In a generic sense of MMAE simulations, MSOFE could be used to run each elemental filter independently from initial-time to final-time, with the MMAE aspects (state and covariance blending, parameter estimation, and etc.) being conducted post-process. This is not, however, really feasible if the MMAE-optimized values are used in calculations which affect the elemental filter calculations during the ensuing propagation and update cycles. With that in mind and although not necessarily an exhaustive list, several types of stochastic adaptive control which are discussed in Chapter 15 of Maybeck's Volume 3 (20) are of interest with regards to MMSOFE.

Closed-loop forms of stochastic adaptive control which use MMAE to estimate states and parameter values optimally require a true MMAE structure as available with MMSOFE. These parameter values are then used in an adaptive control law as shown in Figure 2.1, known as an MMAE-Based Controller. In using this type of controller, $G_C^*[t_i; \hat{a}(t_i^+)]$ is determined as based on the parameter estimate \hat{a} which is assumed to remain constant until the next measurement update. Finally, Maybeck (20) discusses Multiple Model Adaptive Controllers (MMAC) as a type of stochastic adaptive controller which consists of a bank of Linear Quadratic Gaussian (LQG) controllers in place of just the elemental filter of MMAE (see Figure 2.2). Again with MMAC, the MMAE structure is used to calculate probabilities for use as weighting factors to blend the control signals, u_k , from the elemental controllers optimally.

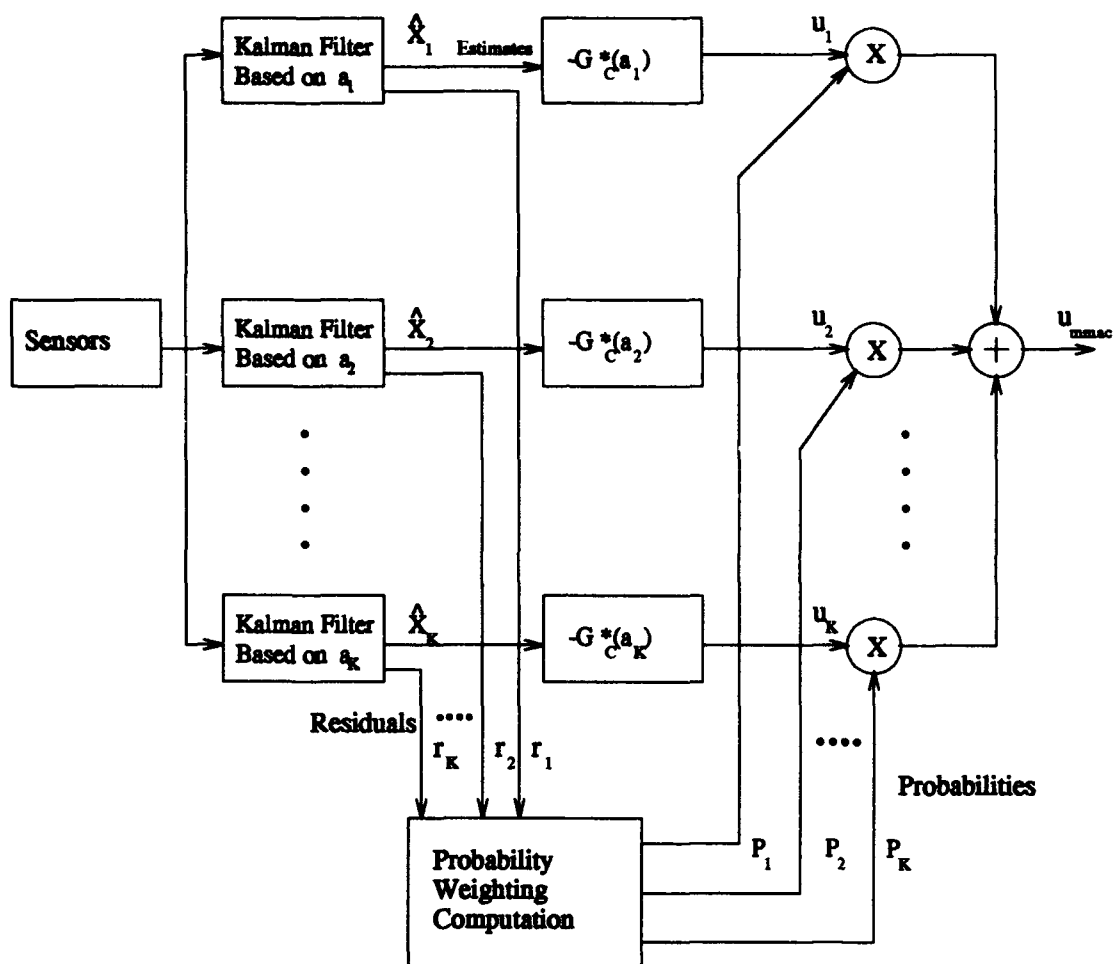


Figure 2.2. Multiple Model Adaptive Controller

2.6 Distributed Kalman Filtering

Distributed Kalman filter (DKF) simulations represent another problem-type which could well be accomplished with a slightly modified MMSOFE. Operationally, DKF would run its various KFs on separate processors with a master (central) filter blending the output of the individual filters. However, for simulation purposes, MMSOFE as conceived for MMAE in this thesis could be modified only slightly to implement a DKF simulation. In such a configuration, even master-filter-calculated values could be fed back to the distributed filters. One type of DKF implementation is depicted in Figure 2.3. In this DKF each local (elemental in MMAE) filter can have a unique model and receive measurements from different local sensors (the same measurements feed into all elemental filters

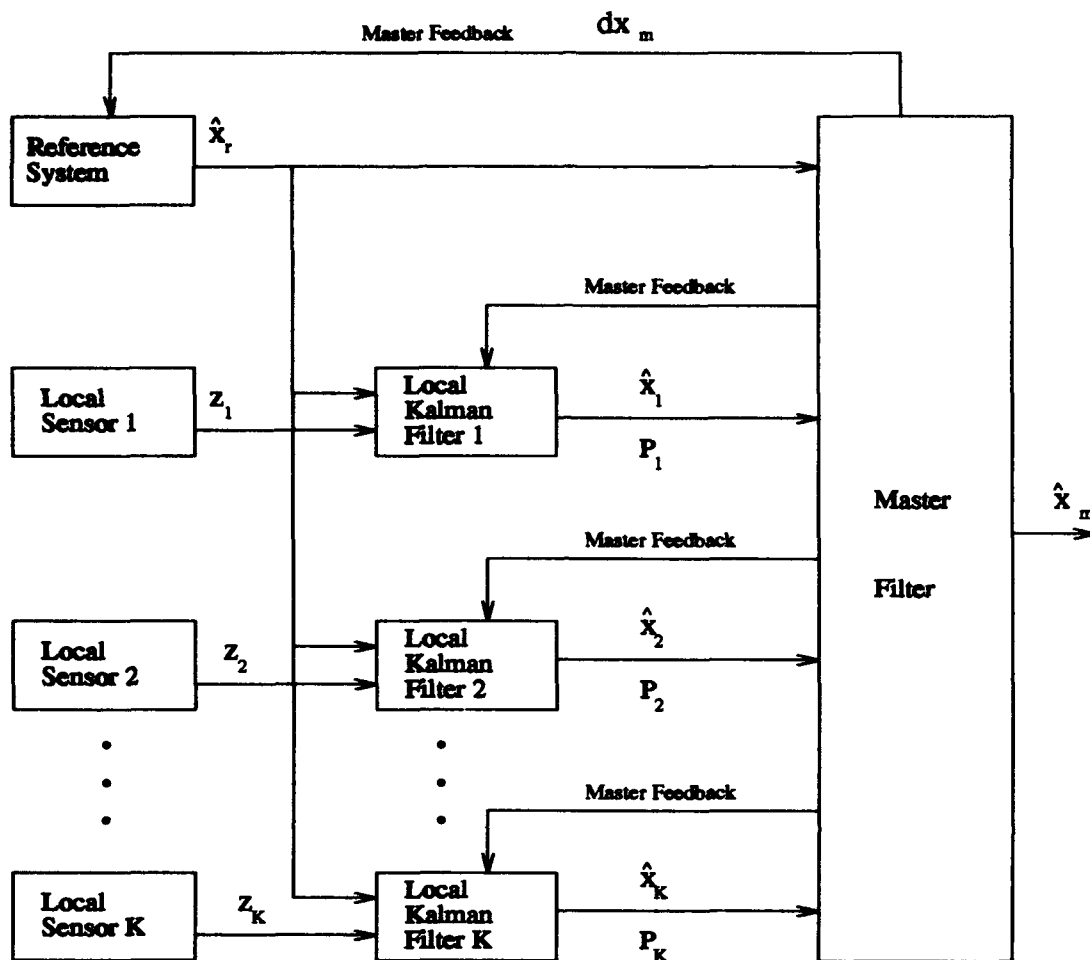


Figure 2.3. One Possible Distributed Kalman Filter Estimator

in MMAE). These local filters calculate the estimates of their unique set of states with related covariances, both of which are forwarded from all of the local filters to the master filter. The master filter's function is to combine or integrate all of these state estimates via some problem-specific combinatorial algorithm or even via a Kalman filter. This master filter may generate some single master feedback or set of master feedbacks which are fed back to the reference system and/or local filters.

2.7 Summary

This chapter has provided the theoretical basis for the remaining chapters. MMAE, FDI techniques, MMAE-based control and MMAC, and distributed Kalman filtering have

been discussed in Chapter II. Chapter III addresses the actual software implementation of MMAE which was developed in this thesis.

III. MMSOFE Development

3.1 Overview

This chapter discusses the aspects of MSOFE which are germane to MMSOFE development, including a brief description of the satellite orbit problem. A cursory discussion of MSOFE and a more detailed description of the MMSOFE adaptation of MSOFE are made, including the actual modifications made to MSOFE. The validation of MMSOFE is discussed next, and finally, MMSOFE capabilities.

3.2 Satellite Orbit Problem

The satellite orbit problem is fully described in Maybeck's Volumes I and II (18, 19) and in the User's Manual for MSOFE (24), but since it is the test problem for MMSOFE development, it warrants some minimal discussion in this thesis. The problem is constructed for MSOFE such that the simulated satellite is already in an orbit with an orbit radius of one "radius unit" from the center of the earth. "Radius units" and "time units" are used as non-dimensional units merely as a simplifying measure. The MSOFE user can then implement a Hohmann orbit transfer as well as modify the initial conditions, noise variances, and choose MSOFE modes and parameters via the input data file MSOFE.IN. The user can specify the times for the Hohmann transfer actions and the ratio of the orbit radii before and after completion of the Hohmann transfer.

As mentioned in Section 3.3, the failures are added into the problem in the subroutine HRZSYS. At the times specified, the failures were induced in HRZSYS with the measurement bias/ramp being added directly to the truth measurement and the measurement noise variance bias/ramp added to the baseline (no-failure) measurement noise variance. Additionally, the Hohmann transfer was not implemented for these simulations during software development, in order to accentuate the performance of the independent elemental filters, the performance of the blended estimation, and that of the probability and parameter calculations.

3.3 MSOFE Description

As mentioned in Chapter 1, MSOFE provides a tool for both Monte Carlo simulations and covariance analysis of linear or extended Kalman filters. The version of MSOFE used for this development was Version 1.5, which was current as of June 1991. The MSOFE FORTRAN code is composed of two source code files, MSOFE.f and USOFE.f. MSOFE.f, containing 59 subroutines, comprises the "core-code" for MSOFE and users should rarely need to access or modify MSOFE.f. USOFE.f, containing 17 subroutines, is the user-interface source code. While some of these USOFE.f subroutines remain as part of the MSOFE code even when they aren't being used for a given application and are just empty shells, they are retained so as to ease the adaptation to other specific filtering problems in which they are required.

In addition to "adapting" USOFE.f for the specific problem, the input and output files which are the other user-interfaces must also be "adapted." MSOFE input and output files are shown in Table 3.1.

Table 3.1. MSOFE Input/Output Files

	File Name	FORTTRAN Unit #	Description
INPUT FILES			
	MSOFE.IN	5	User controls & initial conditions
	FLIGHT	3	Externally produced trajectory
	OLDEND	9	Data from last run
	KFGAIN	11	Kalman gain series
OUTPUT FILES			
	CTOM	2	Continuous-time data
	DTOM	4	Discrete-time data
	MSOFE.OUT	6	Text file
	NEWEND	10	Data for next run
	KFGAIN	12	Kalman gain series
	ERRS	8	Error messages
INPUT/OUTPUT			
	scratch	7	Scratch space

The FORTRAN unit numbers are merely numbers assigned to each particular file when the file is opened, by which FORTRAN refers to files during execution. To explain the files referenced in Table 3.1 further:

MSOFE.IN: Contains controls to direct MSOFE operation and parameters to define the problem and specify initial conditions. MSOFE.IN is broken into five groups: Problem Title, Control Parameters, Printer Plots Specifications, Initial Conditions and User Data.

CTOM & DTOM: Serves as primary data output files for use by plotting or other post-process software.

FLIGHT: Provides the input trajectory data for many MSOFE problems, such as for Vasquez's (36) research in which the "Profile Generator" (PROFGEN) software was used to generate the FLIGHT trajectory file. FLIGHT could alternatively be created via other software available to the user. However, the satellite orbit problem used in MMSOFE development has its trajectory generation software built into USOFE.f code.

KFGAIN: Can be generated by MSOFE and then used as input for future MSOFE runs to specify the precomputed Kalman gain vector during the simulation.

MSOFE.OUT: Generated by MSOFE.f, provides an ASCII record of control values, initial conditions, and any other problem-specific data provided in MSOFE.IN. Could also contain additional problem-specific data as specified by the user in USOFE.f. Also contains error messages and printer plots if requested.

NEWEND & OLDEND: Final condition data is saved to NEWEND at the end of an MSOFE run for use as initial conditions in OLDEND for subsequent MSOFE simulations. The change of name from NEWEND to OLDEND is handled externally by the user.

ERRS: Contains any MSOFE-generated error messages produced during or causing the termination of a run.

Plots of the data in CTOM and DTOM can be made using any of a variety of plotting packages. In this version of MSOFE, CTOM and DTOM were generated in a format compatible as input for MPLOT which is described in the MSOFE user's manual (24). MPLOT was then used to create Matrix_x-compatible data files for plotting with Matrix_x (10).

Because a thorough discussion of MSOFE is available in the User's Manual for MSOFE (24), that effort will not be repeated here, but rather only those parts of MSOFE which were appreciably altered or which bear special significance in MMSOFE development will be discussed in any depth. However, to clarify the modifications to MSOFE, some flowchart-like diagrams are included in this chapter, Figures 3.1 - 3.7. First, the MSOFE top-level executive program, referred to as MAIN, calls several subroutines as shown in Figure 3.1.

These flowchart-like diagrams, Figures 3.1 - 3.7, are obviously not conventional flowcharts. The MMSOFE diagrams, Figures 3.4 to 3.7, were made first and the MSOFE diagrams, Figures 3.1 to 3.3, were made by deleting the non-applicable parts. This results in some of the MSOFE diagrams appearing slightly sparse but allows for better comparisons between the MSOFE and MMSOFE diagrams. The "Note" in these diagrams also requires some additional explanation. These figures are not intended as detailed flowcharts of program code. Rather, they show the flow and order of subroutine calls as they appear in MSOFE or MMSOFE code and the related branch statements. Except for a few loops included for clarity's sake, most of the code is omitted from these MSOFE and MMSOFE diagrams (Figures 3.1 - 3.7). Recalling that the "M" in MSOFE stands for **Multimode**, the "If ____" statements contained in some function blocks or subroutine call-ellipses indicate that action or call statement is conditional on the "If ____" statement being true. Some of these "If ____" conditions vary between passes through the code during a single Monte Carlo "run," or for that matter, for different "runs" of a Monte Carlo simulation. For example, if MSOFE is run in "dual" mode, both covariance analysis and Monte Carlo simulation, the code for both is implemented during the first Monte Carlo "run," but the code specific only to the covariance analysis isn't implemented during subsequent Monte

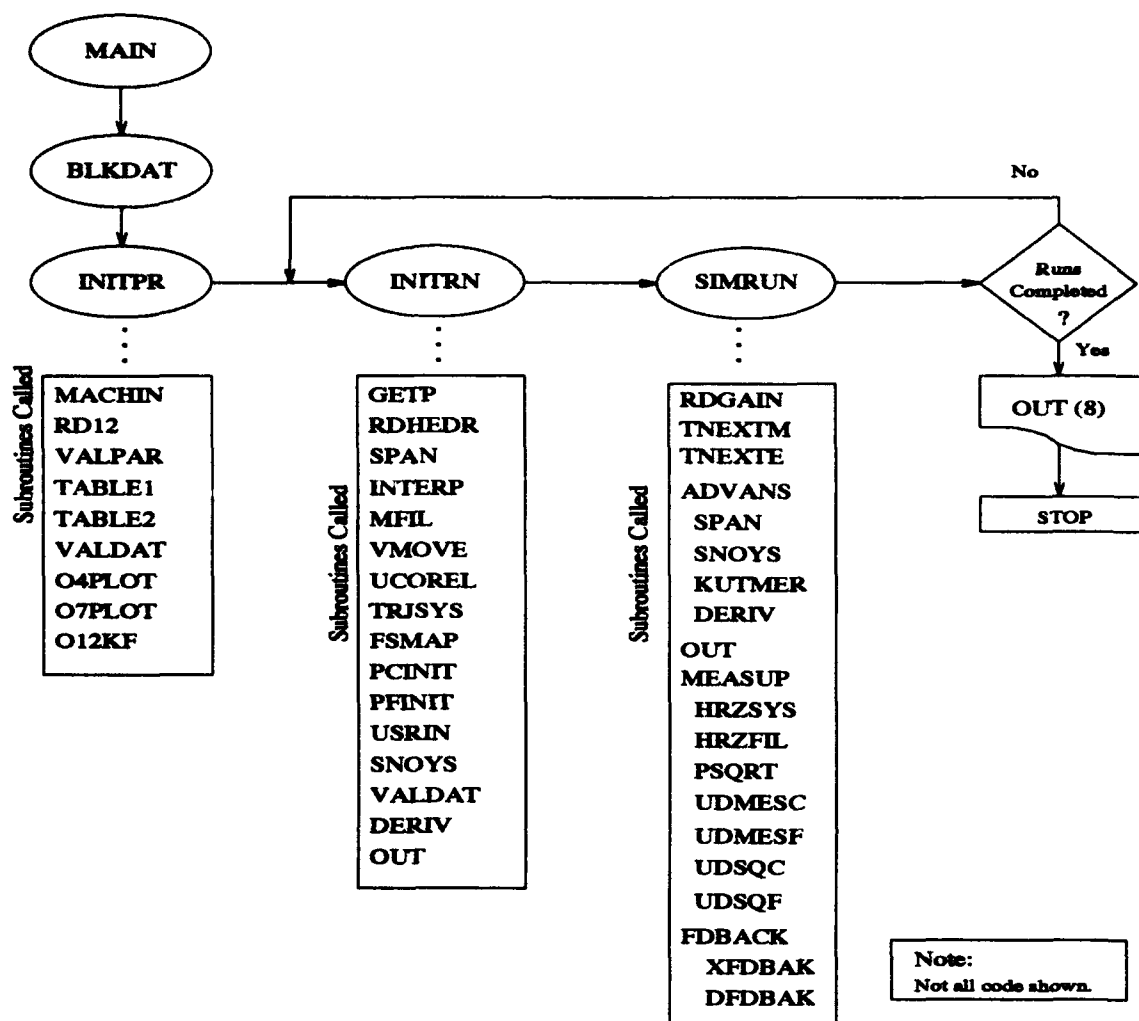


Figure 3.1. MSOFE MAIN Program

Carlo “runs.” Furthermore, since the MSOFE User’s Manual (24) describes the function of all of the MSOFE subroutines, they won’t be described in this thesis.

The blocks in Figures 3.1 and 3.4 containing lists of “Subroutines Called” mean that these subroutines are called by the subroutine that is named in the ellipse directly above the box. The major functions of these subroutines are as follows:

- **BLKDAT:** “Block Data” – Initializes all common areas with “DATA” statements.

- **INITPR:** "Initialize Problem" - Called at the beginning of the simulation to read in problem-specific data from MSOFE_IN and to initialize all variables common to all Monte Carlo runs.
 - **MACHIN:** "Machine" - Computes several computer-specific constants which characterize the computer.
 - **RD12:** "Read Groups 1 & 2" - Reads Group 1 and Group 2 data from MSOFE_IN and writes header information to MSOFE_OUT.
 - **VALPAR:** "Validate Parameters" - Examines parameter from SIZES to ensure they fall within range restrictions.
 - **TABLE1:** "Table 1" - Writes TABLE 1 which contains a list of parameters from the SIZES file and MSOFE_IN Group 2 data to MSOFE_OUT.
 - **TABLE2:** "Table 2" - Writes computer-specific data to MSOFE_OUT.
 - **VALDAT:** "Validate Data" - Examines Group 2 parameters to ensure they fall within range restrictions and to check filter and composite covariance matrices for positive definiteness.
 - **04PLOT:** "Output to Unit #2 & #4" - Writes data to CTOM and DTOM for postrun processing.
 - **07PLOT:** "Output for Plotting" - Executive routine for making printer plots of user-specified variables.
 - **012KF:** "KF Output to Unit #12" - Writes the Kalman gain to the file KF-GAIN.
- **INTRN:** "Initialize Run" - Called at the beginning of each simulation run to read in run-specific data and to initialize run-specific variables.
 - **GETP:** "Get P (covariance)" - Reads MSOFE_IN to acquire the initial values for the filter, filter error, and truth covariance matrices.
 - **RDHEDR:** "Read Header" - Reads the header of the FLIGHT trajectory file.

- SPAN: "Span T" - Reads the data from FLIGHT trajectory file and manipulates data so the simulation times are synchronized with trajectory data.
 - INTERP: "Interpolate" - Interpolates trajectory data to the time series being used.
 - MFIL: "Matrix Fill" - Fills a matrix with a scalar values.
 - VMOVE: "Vector Move" - Copies one vector into another.
 - UCOREL: "Correlated Gaussian Sample Vector" - Constructs a Gaussian sample vector with zero mean.
 - TRJSYS: "Trajectory, System" - Calculates user-specified trajectory data.
 - FSMAP: "Filter/System Mapping" - Computes the matrix AFS which maps truth model states into equivalent filter states.
 - PCINIT: " P_C Initialize" - Initializes the composite covariance matrix.
 - PFINIT: " P_F Initialize" - Initializes the filter covariance matrix.
 - USRIN: "User Input" - Reads data as specified by the user.
 - SNOYS: "System Noise" - Provides random noise to the truth state vector.
 - VALDAT: See description under INITPR above.
 - DERIV: "Derivative" - Called to initialize the derivatives of the truth and filter states, as well as of the filter and composite covariances.
 - OUT: "Output" - Controls the outputs which are specified.
- SIMRUN: "Simulation Run" - The top-level controller for the time cycles of simulation runs.
 - RDGAIN: "Read Gain" - Reads the next measurement time and the Kalman gain from the KFGAIN file.
 - TNEXTM: "Time of Next Measurement" - Calculates the next measurement time for each measurement type.
 - TNEXTE: "Time of Next Event" - Determines the time of the next event.

- ADVANS: "Advance" - Controls the propagation between events via numerical integration.
- SPAN: See description under INITRN above.
- SNOYS: See description under INITRN above.
- KUTMER: "Kutta Merson" - Integrates a set of first-order ordinary linear or non-linear differential equations.
- DERIV: See description under INITRN above.
- OUT: See description under INITRN above.
- MEASUP: "Measurement Update" - Performs the measurement update for each scalar measurement one at a time.
 - HRZSYS: "System H, R, Z" - Supplies the truth measurement (Z), the truth measurement noise variance (R), and the partial derivatives of the true measurement with respect to the filter states and the system states (H).
 - HRZFIL: "Filter H, R, Z" - Supplies the filter-predicted measurement (Z), the filter measurement noise variance (R), and the partial derivatives of the filter measurement with respect to the filter states and the system states (H).
 - PSQRT: "Square Root of the System Covariance" - Calculates the square root of the covariance matrix in terms of the unitary (U) and diagonal (D) matrices.
 - UDMESC: "Composite UD Measurement" - Performs the measurement update of the composite covariance U and D matrices.
 - UDMESF: "Filter UD Measurement" - Performs the measurement update of the filter covariance U and D matrices.
 - UDSQC: "Composite UD Squared" - Forms the composite covariance matrix in upper triangular form from the U and D factors.
 - UDSQF: "Filter UD Squared" - Forms the filter covariance matrix in upper triangular form from the U and D factors.

- **FDBACK:** "Feedback" – Updates the filter and system states and/or the composite covariance matrix with regard to the discrete feedback correction.
 - **XFDBAK:** "State Feedback" – Updates truth and filter states with discrete feedback correction.
 - **DFDBAK:** "D Feedback" – Determines the matrices used to pre-multiply the truth or filter state vectors to provide discrete corrections to the states.

BLKDAT (block data) contains the default initialization **DATA** statements to initialize the values of variables, in contrast to **PARAMETER** statements which set the values of parameters. The **BLKDAT** "call" is handled as "**EXTERNAL BLKDAT**", not as a normal subroutine call. Parameter values cannot be changed in the program but, of course, variables can be changed. Conversely, parameters can be used to dimensionalize vectors and matrices in declaration statements but variables cannot. This "**EXTERNAL BLKDAT**" statement has the effect of placing all the **DATA** statements contained in **BLKDAT** right there where the **EXTERNAL** statement is found without having all the **DATA** statements actually present in the top-level executive routine.

The **MSOFE** subroutines **INITPR**, **INITRN**, **SIMRUN**, and **OUT** are all called from **MAIN** and are all germane to **MMSOFE** development. **INITPR** is the subroutine which initializes the variables and flags for the entire **MSOFE** simulation and is only executed once for **MSOFE**. **INITRN** initializes some variables and flags for each run in a Monte Carlo **MSOFE** simulation and is executed at the beginning of each run. **SIMRUN**, the **MSOFE** second-level executive subroutine, calls a number of other subroutines, as shown in Figure 3.2. **SIMRUN** is responsible for timing and scheduling propagations, measurement updates, feedback, and judiciously calling **OUT** with the appropriate argument. The **OUT** subroutine controls **MSOFE** scheduled output, special prints, and a few other special functions. The **OUT** integer argument, **IOUT**, shown in the diagrams with the **OUT** calls, specifies the event which has occurred and determines **OUT**'s internal flow.

With respect to **MMSOFE** development, the call to **MEASUP** (Figure 3.3) is the most notable subroutine call in **SIMRUN**. It is the most significant because, in **MMSOFE**, **MEASUP** is where nearly all of the calls to the **MMSOFE** **MMAE** subroutine are made

(see Figures 3.4 – 3.7). The MMSOFE MMAE subroutine, the code of which is presented in Figure A.1 and which is discussed in Section 3.4.2.3, performs nearly all of the multiple model related calculation and also writes to the output file, BLCNTOM (data file containing the blended or MMAE-related output). So, the filename BLCNTOM was derived from CTOM, the MSOFE output file, with BL = blended and N merely referring to N files for N elemental filters.

Subroutine SIMRUN handles the time progression of measurements and other events for MSOFE (TNEXTM, TNEXTE) and the subroutine calls for propagation (ADVANS), measurement update (MEASUP), subroutine calls for the application of discrete feedback (FDBACK), and the outputs at appropriately scheduled times (OUT). The subroutine SIMRUN performs all of these functions for all Monte Carlo runs before returning control to the MSOFE MAIN program.

The MEASUP subroutine, which is called by SIMRUN, controls the calculations related to the measurement update by calling subroutines HRZSYS and HRZFIL. In MSOFE's subroutine MEASUP, HRZSYS and HRZFIL are the subroutine calls which are most pertinent to MMSOFE development. These two subroutines reside in USOFE.f and are therefore intended for user interface. They build the partial derivative matrices which are: derivative of the true measurement with respect to the truth states (HSYSS), derivative of the true measurement with respect to the filter states (HSYSF), derivative of the filter measurement with respect to the truth states (HFILS), and derivative of the filter measurement with respect to the filter states (HFILS). These are used in MEASUP to build the measurement observation vectors. HRZSYS and HRZFIL also build the truth and filter measurements respectively, including measurement noise addition for the truth measurement. For the orbit estimation problem as used for this thesis, HRZSYS is where the failures given in Table 3.1 were instilled into the true range measurement or true measurement noise variance. Likewise, HRZFIL was used to add the elemental filter-assumed measurement signal biases or measurement noise strength alterations for both MSOFE and MMSOFE. Subroutine MEASUP also controls the calls to subroutines UDMESC and USMESF which perform the $\mathbf{U} - \mathbf{D}$ covariance factorization calculations for updating the covariances for the measurement updates, and it controls the calls to the UDSQC and

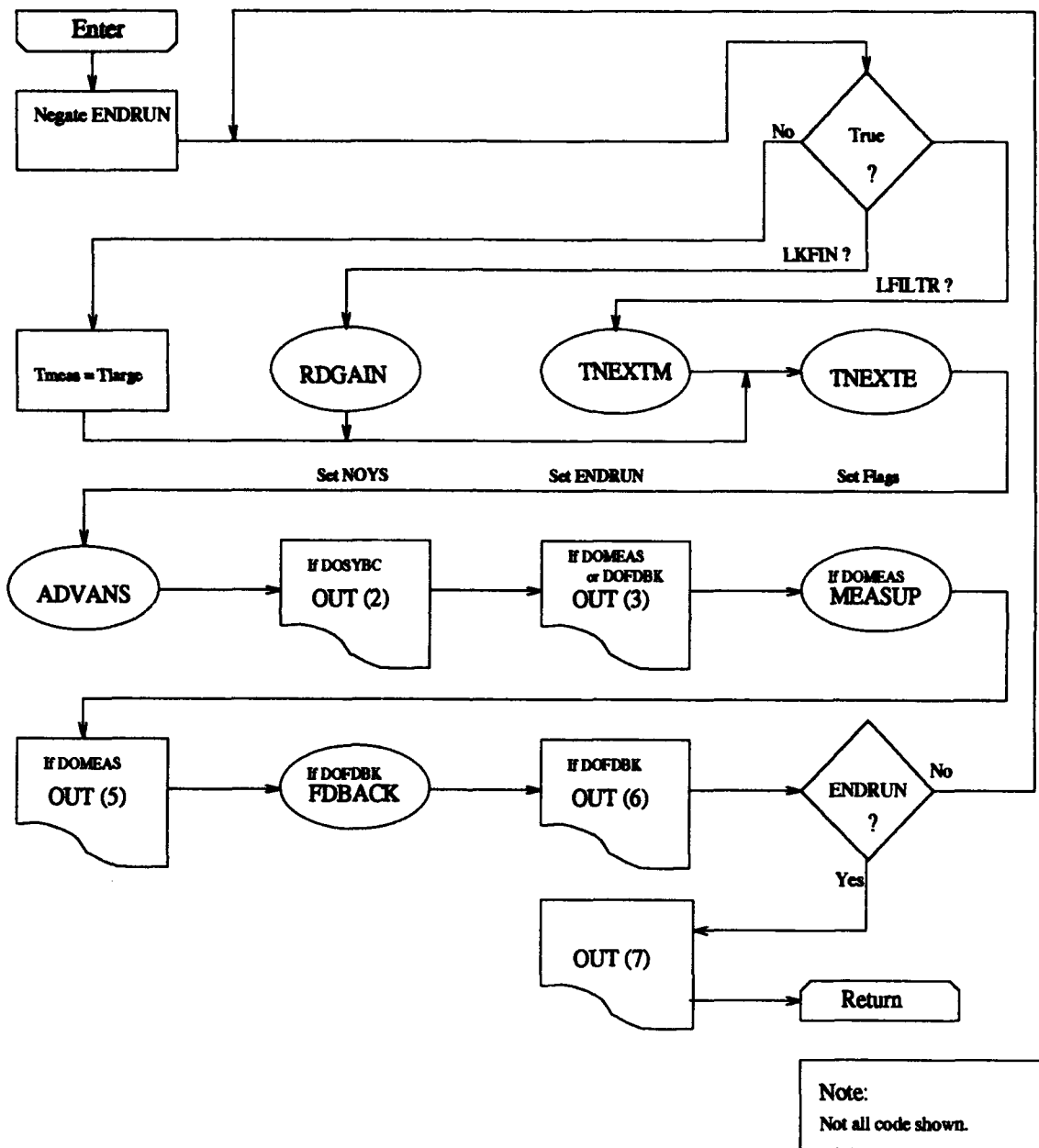


Figure 3.2. MSOFE SIMRUN Subroutine

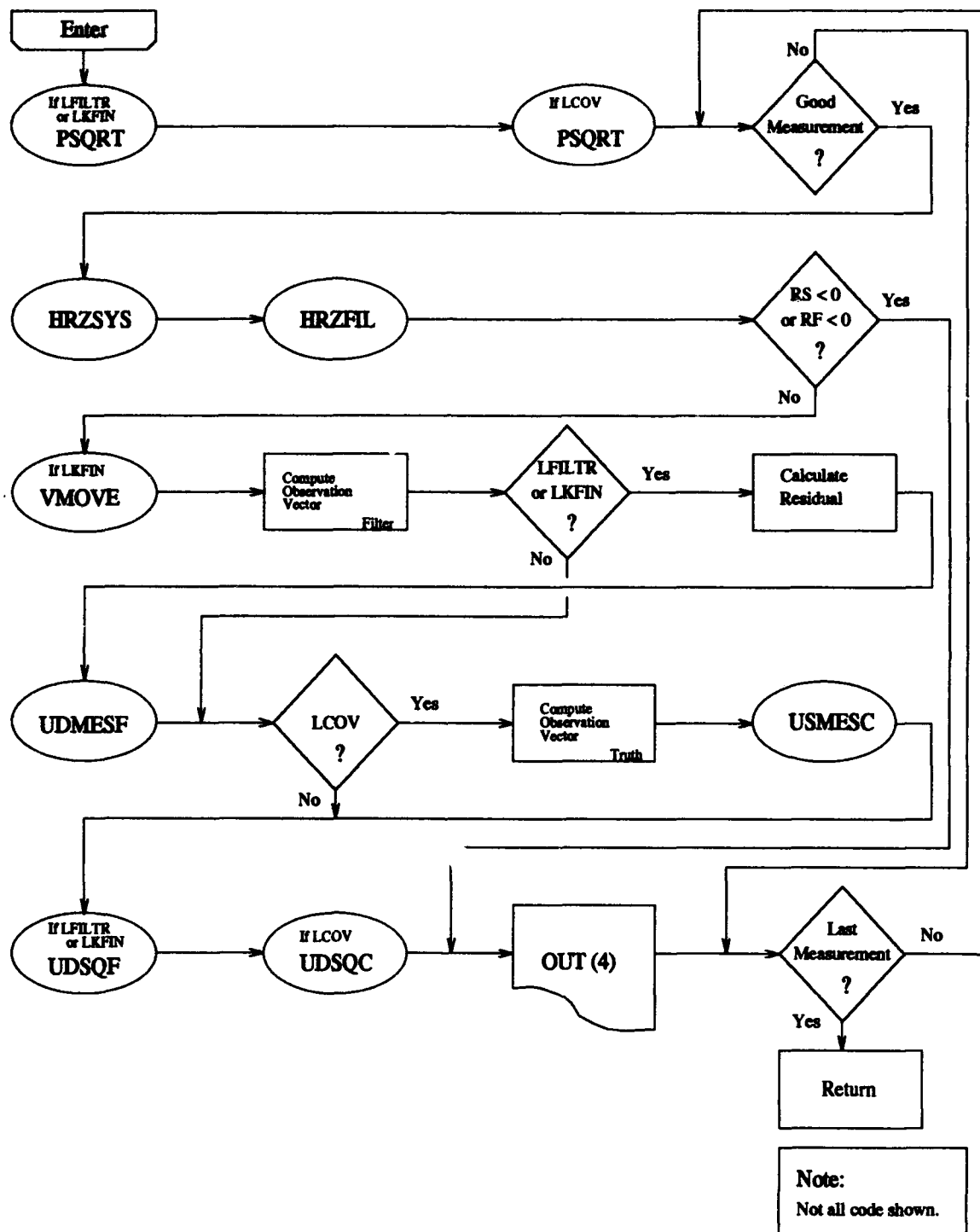


Figure 3.3. MSOFE MEASUP Subroutine

3.4 *MMSOFE Adaptation from MSOFE*

UDSQF subroutines which form the appropriate covariance matrix elements and stores them in a vector containing the upper triangular matrix elements.

After it had been determined that SIMRUN would be the top-level executive subroutine for MMOFE, the methodology followed for MMOFE development was as discussed in Section 1.6.4. First, The MMOFE input and output files will be described. Then, the new subroutines developed for MMOFE (EMFIL, INVERT, MMAE, and OPFILE) are discussed, followed by discussions of the MMOFE versions of subroutines modified from MSOFE subroutines (MAIN, SIMRUN, and MEASUP) and the MMATH subroutine which was borrowed and modified from Version 1.5 of MSOFE.f, current in February 1993 and named CSOFE.f. CSOFE.f is actually the version of MSOFE described in the User's Manual for MSOFE (24). Finally the "vectorization" of MSOFE scalar variables (making elemental filter-specific or blended output-specific variables) and "matricization" of MSOFE vectors will be discussed, but both will be referred to as "vectorization".

3.4.1 Input/Output Files. Because more input and output files are required for MMOFE than for MSOFE (see Table 3.1), a new subroutine, OPFILE (Section 3.4.2.4 and Figure A.2), was devised to accommodate automatically opening and appropriately naming the input and output files. OPFILE assigns names and FORTRAN unit file numbers to the MMOFE input and output files as required by the number of elemental filters being utilized. The names and unit numbers of the files opened by OPFILE correspond somewhat to the files used by conventional MSOFE. The following table, Table 3.2, shows that correspondence with MSOFE and indicates the file naming convention used by MMOFE. There are several points of explanation necessary with regard to Table 3.2. The "xx" refers to the total number of elemental filters used. If the computer being used could handle such a large problem quickly enough, OPFILE would allow MMOFE to process an MMAE simulation with up to 98 elemental filters (i.e. $xx = 98$). BLCNTOM is the output file for the MMAE blended outputs, probabilities, and parameter estimates, and all of their related statistics. BLDSTOM was not used for this thesis effort, but it is opened

Table 3.2. MMSOFE Input/Output Files

	MISOFE File Name	MISOFE File Unit #	MMSOFE File Names	MMSOFE File Unit #s
INPUT FILES				
	MISOFE.IN	5	01INPUT - xxINPUT	501 - 5xx
			& BLINPUT	5xx+1
	FLIGHT	3	unchanged	3
	OLDEND	9	OLEND	9
	KFGAIN	11	unchanged	11
OUTPUT FILES				
	CTOM	2	01CNTOM - xxCNTOM	201 - 2xx
			& BLCNTOM	2xx+1
	DTOM	4	01DSTOM - xxDSTOM	401 - 4xx
			& BLDSTOM	4xx+1
	MISOFE.OUT	6	01MDATA - xxMDATA	601 - 6xx
			& BLMDATA	6xx+1
	NEWEND	10	unchanged	10
	KFGAIN	12	unchanged	12
	ERRS	8	01ERRRS -	801 -
			xxERRRS	8xx
			& BLERRRS	8xx+1
INPUT/OUTPUT				
	scratch	7	01scratch -	701 -
			xxscratch	7xx
			& BLscratch	7xx+1

by MMSOFE and could be used by an MMSOFE user. The distinctions "C = continuous" in CTOM and the "D = discrete" in DTOM are determined by the header information, not by the OPEN statement, and those distinctions are maintained in MMSOFE _CNTOM and _DSTOM. Similarly, the BLINPUT, BLMDATA and BLERRRS files are not used but are also opened and available for use. The BLINPUT file was specifically left for use as an input file for data which might be required for some of the other applications such as MMAE-based stochastic based control, MMAC, or DKF.

One other caveat to the MMSOFE file structure has relevance only when MMSOFE is used to simulate a single filter. When there is only one filter instead of multiple filters, the "prefix" will be absent from all filenames and the "BL" files aren't utilized either. For example, the output files _CNTOM and _DSTOM would be just CNTOM and DSTOM, and there would be no BLCNTOM nor BLDSTOM. Similarly, _INPUT would be just INPUT and there would be no BLINPUT. This feature was designed to minimize memory use when performing single filter simulations.

The _INPUT files are identical to MMSOFE_IN except for several parameters added at the end of the _INPUT files to ease the modification of problem-specific and/or elemental filter-specific variables. This is not to say, however, that all of the values input via the _INPUT files can be different for every filter. An MMSOFE user might, for instance, want to vary the number of measurements (NM) between different elemental (local) filters in a DKF application or vary the initial (TMI) or final times (TMF) of measurement availability for some application, but these variables which are read from the _INPUT files would first require vectorization. While MMSOFE could be further modified to facilitate an implementation for some particular application, because MMSOFE reads from the _INPUT files in numerical order, MMSOFE will retain the last value read for any variable which hasn't been vectorized, namely the value from the xxINPUT file (highest numbered _INPUT file). Those variables which have been vectorized are discussed in Section 3.4.4. For the satellite orbit problem test-case, the variables added at the end of the _INPUT files include:

1. **RNERR** is the number corresponding to the failure type implemented in **HRZSYS** and is a scalar, remaining constant for all elemental filters.
2. **MINPROB** is the lower bound value for the probabilities to prevent the "locked in" situation described in Section 2.3 and is also a scalar.
3. **ZFBias** is a vector of values containing the hypothesized measurement signal bias magnitudes corresponding to the individual elemental filters.
4. **RFNOIS** is a vector of values corresponding to the hypothesized measurement noise variance magnitudes applied to the different elemental filters.

The calls to subroutine **OPFILE** to generate the **xx** files corresponding to **MSOFE**'s files **FLIGHT**, **OLDEND/NEWEND**, and **KFGAIN** were included in **MMSOFE**, but they weren't implemented in this thesis because none of these four files were required for the orbit estimation problem. If an **MMSOFE** user wanted to conduct a study using such files specific to each elemental filter, these call statements could be easily activated.

3.4.2 New Subroutines. The two completely new subroutines are included in Appendix A; **MMAE** in Figure A.1 and **OPFILE** in Figure A.2.

3.4.2.1 EMFIL Subroutine. The **EMFIL** subroutine computes the **MMAE** blended state estimation error vector, $E_{blended}$ given by:

$$E_{blended} = \hat{X}_{Blended} - AFS \cdot X_{Truth} \quad (3.1)$$

where $\hat{X}_{Blended}$ represents the **MMAE**-blended state estimates, **AFS** is the filter/system mapping matrix computed by subroutine **FSMAP** in **USOFE.f**, and X_{Truth} is the true state vector. **EMFIL** was adapted from the **MSOFE.f** subroutine **EFFIL** which finds the state estimation error for the single filter and for the elemental filters in **MMSOFE**.

3.4.2.2 INVERT Subroutine. The **INVERT** subroutine takes the inverse of a square matrix of any dimension using the method of Gaussian elimination. It was written in 1988 by students at the Air Force Institute of Technology advised by Dr. Peter Maybeck on projects sponsored by the Air Force Weapons Laboratory at Kirtland Air Force Base,

NM. This author takes no credit for INVERT but did check its function to ensure it was accurate in its calculations.

3.4.2.3 MMAE Subroutine. The subroutine MMAE, contained in entirety in Appendix A in Figure A.1, was written to perform nearly all of the MMAE-related calculations in MMSOFE. The argument MOUT which is used in calling MMAE, determines what MMAE's function will be as follows:

1. Initialize the problem at the start of the simulation (Calls INITPR).
2. Initialize the runs. INITPR is called, several parameter-related variables are initialized, and the probabilities for all the elemental filters are initialized to one divided by the number of elemental filters.
3. Copy the vectors saved from the last propagation and update cycle for an elemental filter into working vectors for the current cycle. The variables saved are as follows:
 - AFS = FILTER/SYSTEM MAPPING MATRIX
 - XF = FILTER STATE VECTOR
 - XS = SYSTEM TRUTH STATE VECTOR
 - EF = FILTER/SYSTEM ERROR VECTOR
 - KF = FILTER KALMAN GAIN VECTOR
 - CF = FILTER CONTINUOUS FEEDBACK CONTROL MATRIX
 - DF = FILTER DISCRETE FEEDBACK CORRECTION MATRIX
 - FF = FILTER FUNDAMENTAL DYNAMICS MATRIX
 - QF = FILTER PROCESS NOISE POWER DENSITY (STRENGTH) MATRIX
 - PC = COMPOSITE COVARIANCE VECTOR, UPPER TRIANGLE
 - PF = FILTER COVARIANCE VECTOR, UPPER TRIANGLE
 - PE = FILTER ERROR COVARIANCE VECTOR, UPPER TRIANGLE
 - YT = TRAJECTORY VECTOR AT TIME T

As discussed in Section 3.4.4 with regard to vectorization, not all of these would have strictly required saving for the satellite orbit problem, but it was deemed prudent in order to make future adaptations easier.

4. Save the vectors shown in #3 directly above for the next iteration.
5. Write the following MMAE-related values to output file designated for the MMAE-related data (BLCNTOM):
 - T = Simulation time.
 - XFMAT = State estimates for all elemental filters
 - XFM = Blended (MMAE) state estimates.
 - PFMM = Blended state covariance matrix.
 - PFMMD = Diagonals of the blended-state covariance matrix.
 - EM = Blended state estimation error.
 - PROB = Probabilities of the elemental filters.
 - PAREST = Parameter estimate.
 - EA = Parameter estimation error.
 - EP = Diagonals of the parameter estimation error.
 - TRUPAR = True value of the parameter from subroutine HRZSYS.
6. Save $\hat{\mathbf{x}}_{filter}^-$ for residual calculations and save $\hat{\mathbf{P}}_{filter}^-$ into matrix form for calculation of the covariance of the blended residuals.
7. Perform the following MMAE calculations required for each elemental filter.
 - Save the current elemental filter's state estimates and upper triangular covariance vector values into the storage matrices, XFMAT and PFMAT.
 - Calculate the measurement residual vector.
 - Calculate the covariance matrix (ALPHAM) of the measurement residual vector and find the inverse of ALPHAM.

- Calculate the exponential value used to calculate the density.
 - Calculate the coefficient value which precedes the exponential used to calculate the density.
8. Perform calculations for Bayesian multiple model adaptive estimation only when the last elemental filter has been complete for each propagation/update cycle.
- Calculate the denominator of the conditional probabilities.
 - Evaluate the conditional probabilities.
 - Check the conditional probabilities against the minimum probability value set in the `_INPUT` files. If any are smaller:
 - Set them equal to the minimum probability value.
 - Recalculate the denominator of the new adjusted conditional probabilities.
 - Re-evaluate the conditional probabilities.
 - Calculate the blended (MMAE) state estimates using the probabilities as weighting factor on the individual state estimates from the elemental filters.
 - Calculate the covariance matrix for the blended state estimates (see Equation (2.31)).
 - Compute the blended state estimation error from the true state values.
 - Calculate the estimates of the parameters.

3.4.2.4 OPFILE Subroutine. The subroutine, `OPFILE`, was written to open and name the input and output files in `MMSOFE` automatically. The `OPFILE` subroutine is contained in its entirety in Appendix A, Figure A.2. `OPFILE` assigns names and FORTRAN unit file numbers to the `MMSOFE` input and output files as required by the number of elemental filters being utilized. The naming convention is that the normal ASCII filename has a two digit numerical prefix corresponding to the elemental file number. An extra few files beginning with "BL" instead of the elemental filter number are also opened, with "BL" representing "BLended". The FORTRAN unit file numbers are assigned with an even hundred's value representing the type of file with a value added which corresponds

to the elemental filter number, or one more than the total number of elemental filters for the "BL" files. For example, if there were three elemental filters,

- The MSOFE CTOM file had a FORTRAN file unit number = 2
- The MMSOFE files corresponding to CTOM would be 01CNTOM, 02CNTOM, 03CNTOM, and BLCNTOM.
- The FORTRAN unit file numbers for these _CNTOM files would be 201, 202, 203, and 204, respectively.

OPFILE was constructed to be flexible to the extent that, if the computer being used could handle such a large problem quickly enough, OPFILE would allow MMSOFE to process an MMAE simulation with up to 98 elemental filters (i.e. $xx = 98$). These file names as corresponding to MSOFE file names are shown in Table 3.2 on page 3-9, and the vectorized variables in OPFILE are discussed in Section 3.4.4.

3.4.3 Versions of MSOFE Subroutines.

3.4.3.1 MAIN Subroutine. In MMSOFE, the MAIN program is no longer the top-level executive routine, but rather that distinction has been relegated to SIMRUN. In fact, MSOFE's MAIN subroutine (see Figure 3.1) was stripped of nearly all of its functions in developing MMSOFE MAIN (see Figure 3.4). In comparing the two versions of the MAIN subroutine, as shown in these Figures, this difference is apparent. The BIKDAT "call" is handled as an "External" call, not as a subroutine call in both versions. The subroutine calls in MSOFE MAIN to subroutines INIPR, INITRN, and OUT have been moved down into SIMRUN and the new MMSOFE subroutine (MMAE) will also be called from SIMRUN. The only true subroutine call remaining in the MMSOFE MAIN is the call to SIMRUN, with a loop around the "CALL SIMRUN" statement to facilitate incrementing the run number for Monte Carlo simulations. So, in comparing Figure 3.1 (MSOFE MAIN) to Figure 3.4 (MMSOFE MAIN), the INIPR, INITRN, and OUT subroutines and the one new subroutine MMAE are the only changes to the boxed list of subroutines that are called by SIMRUN.

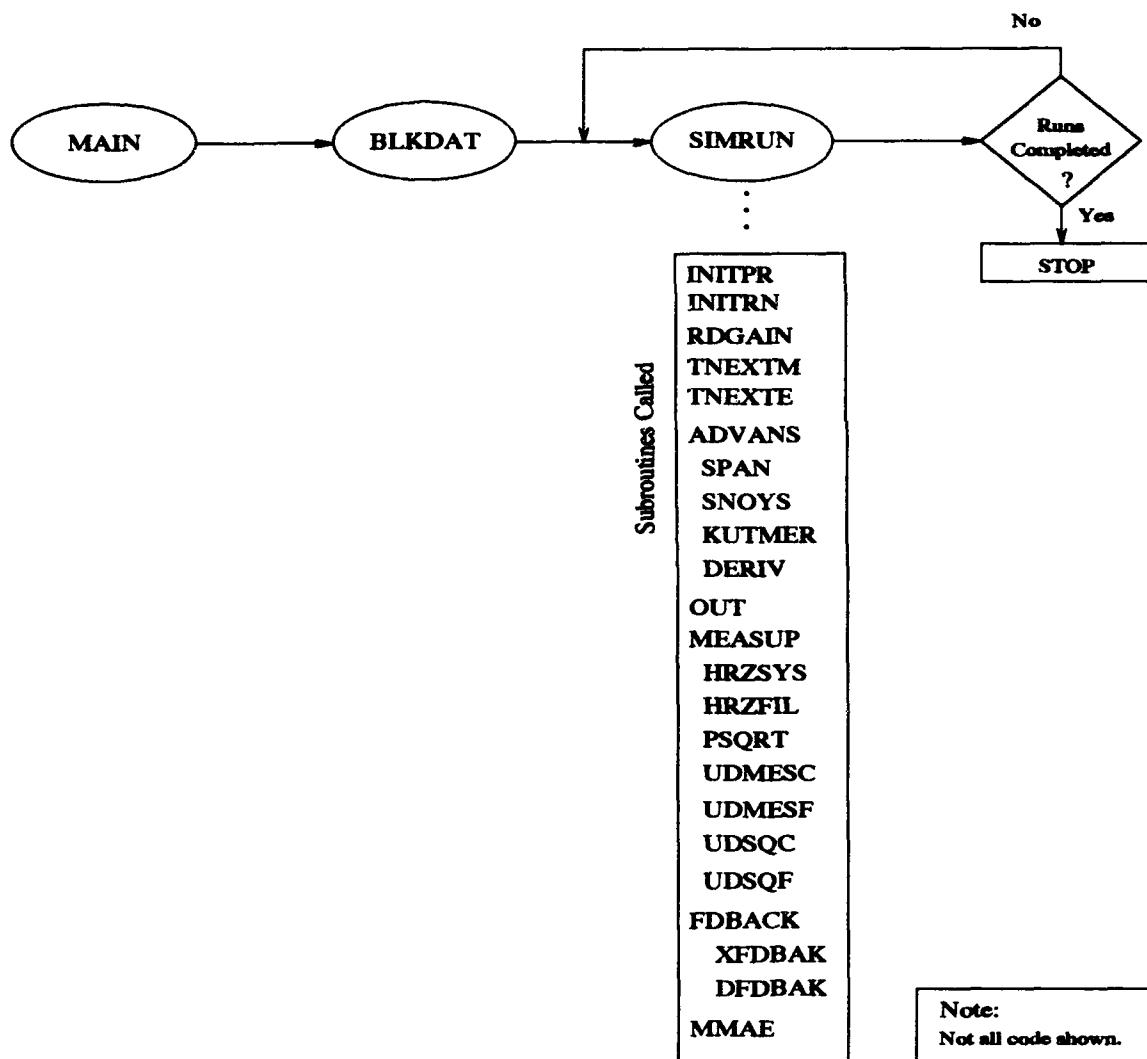


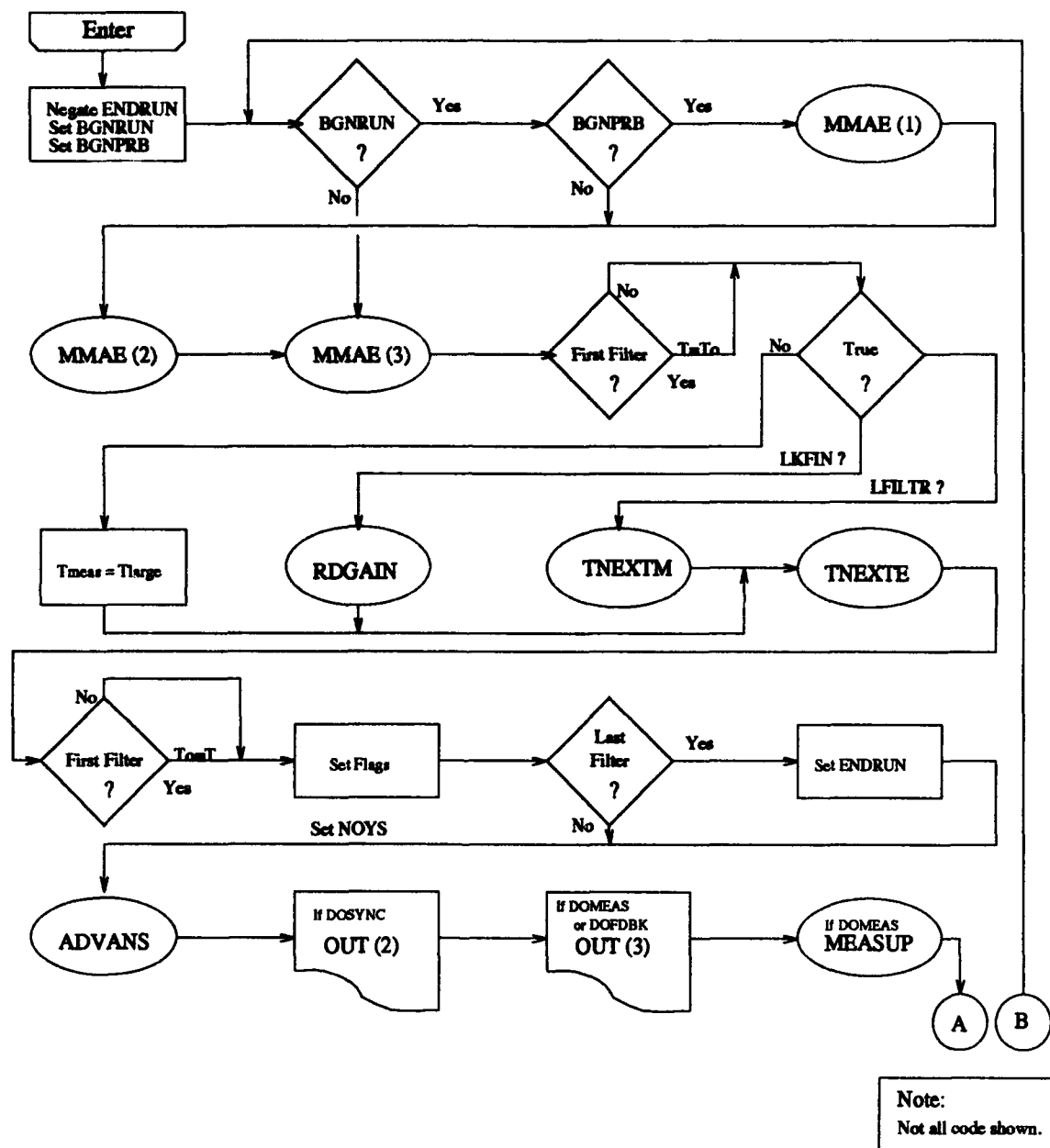
Figure 3.4. MMSOFE MAIN Program

3.4.3.2 SIMRUN Subroutine. Beside being used as the MMSOFE top-level executive subroutine, SIMRUN is also required to handle the coordination of which elemental filter is currently being processed. The program flow for the new MMSOFE SIMRUN subroutine is shown in Figures 3.5 and 3.6. This is accomplished primarily via calls to the new *MMAE* subroutine, monitoring the elemental filter number, and using the flags, BGNRUN and BGNPROB. BGNPROB stays "true" until all elemental filters are initialized at the start of a simulation. BGNRUN remains "true" while initializing all elemental filters at the start of each Monte Carlo run. Comparing Figure 3.2 to Figures 3.5 and 3.6, other changes from MSOFE to MMSOFE include:

1. MMSOFE includes time manipulations to ensure a common time progression for all elemental filters.
2. In MMSOFE, the ENDRUN flag is set to "true" only when the last elemental filter has finished the last run.
3. The calls to subroutine OUT are controlled both during a run and at the end, OUT(8), of the simulation for each elemental filter in MMSOFE. Subroutine OUT(8) was called from MAIN in MSOFE.
4. MMSOFE's SIMRUN decrements the run counter (IRUN) between elemental filters during run initializations. In both MSOFE and MMSOFE, IRUN is incremented in subroutine INITRN and therefore, it must be decremented between passes through INITRN for each elemental filter.

Therefore, it should be obvious by comparing the SIMRUN subroutine from MSOFE versus SIMRUN from MMSOFE that the basic structure of MSOFE remains intact in MMSOFE.

3.4.3.3 MEASUP Subroutine. The MEASUP modifications can be seen by comparing MSOFE MEASUP depicted in Figure 3.3 with MMSOFE MEASUP as shown in Figure 3.7. MEASUP in MMSOFE makes three calls to subroutine MMAE. The additions to MSOFE MEASUP are the three calls to the MMAE subroutine, some initializations at the beginning of MEASUP, and the vectorizations discussed in Section 3.4.4.



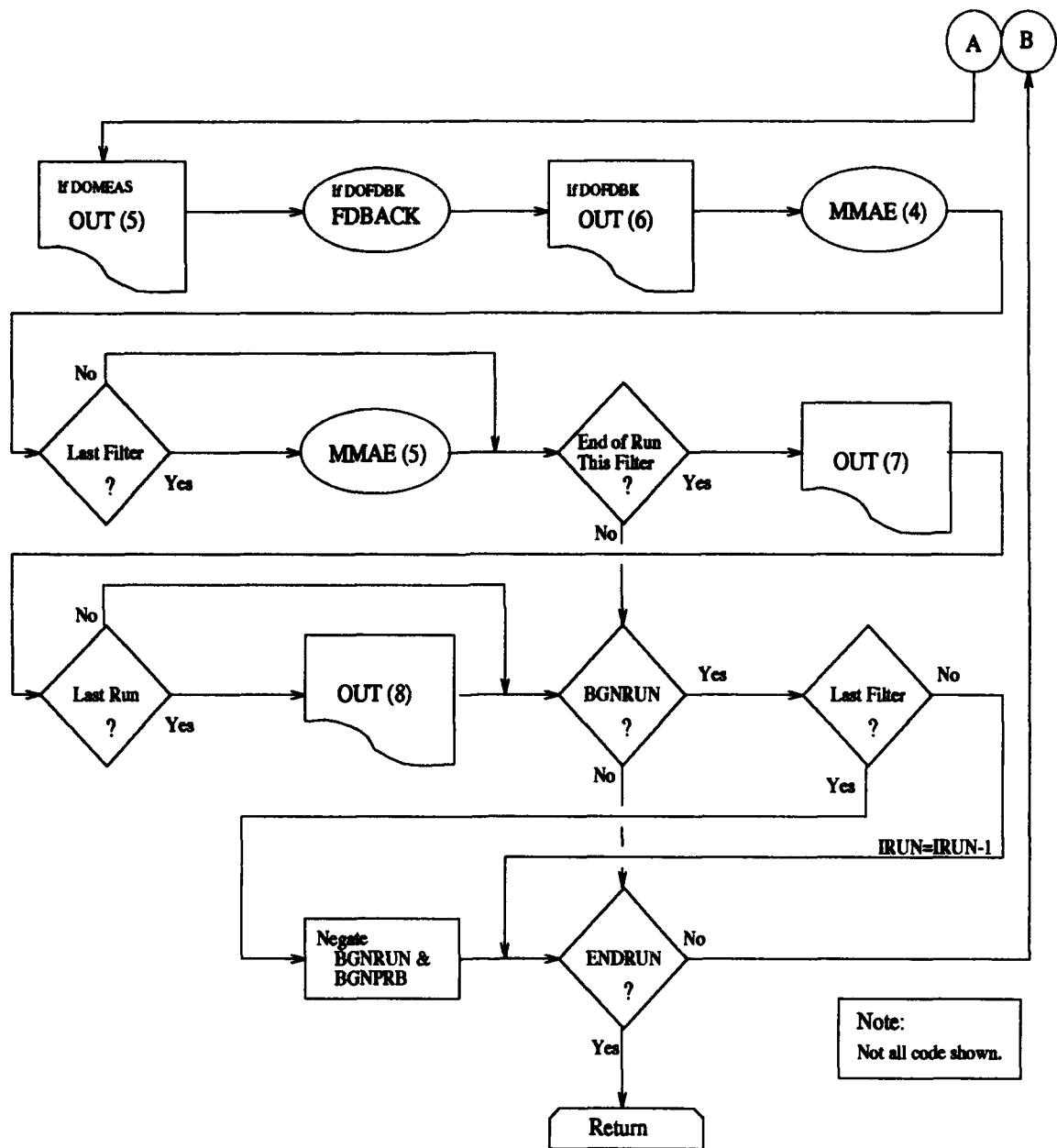


Figure 3.6. MMSOFE SIMRUN Subroutine (Part B)

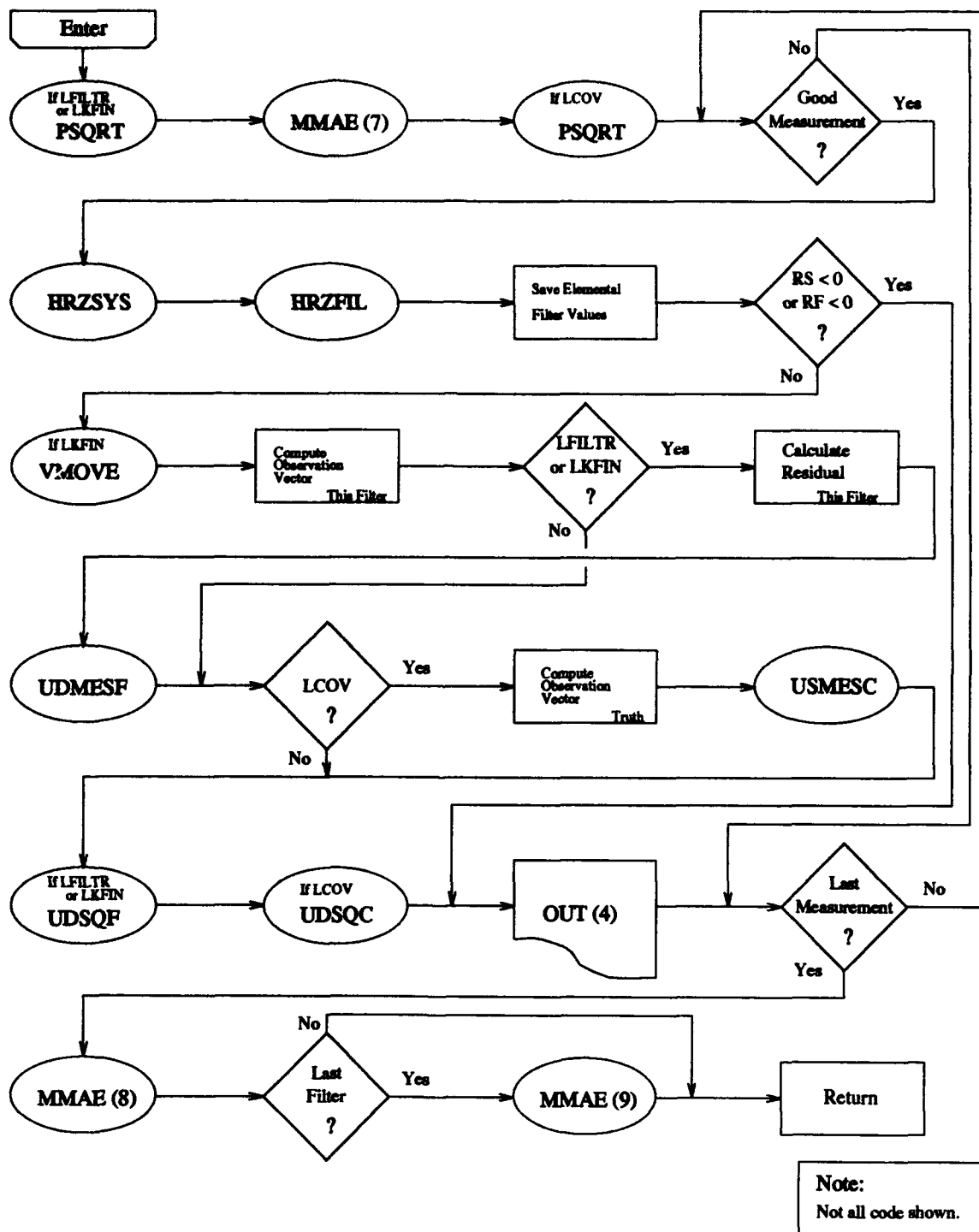


Figure 3.7. MMSE MEASUP Subroutine

Once again, the changes to MSOFE are fairly minimal, preserving the MSOFE structure and using it for calculating each independent elemental filter's values in turn. Because the changes were minimal, no diagram of subroutines HRZSYS nor HRZFIL was made, but those changes were important from an MMAE point of view. Besides inserting the failure signals in HRZSYS and HRZFIL, the respective truth and filter measurements or measurement noise variances were loaded into matrix form for use in MMAE calculations in subroutine MMAE. Additionally, in HRZFIL, the filter derivative vector, HFILF, was also saved into a matrix for MMAE calculation use. Because the calculations were done for each elemental filter independently and in turn during a propagation/measurement update cycle, these matrices were re-used for the elemental filters instead of having to be saved for each elemental filter until MMAE calculations could be accomplished.

3.4.3.4 MMATH Subroutines. The MMATH subroutine is merely a very useful linear algebra utility subroutine which simplifies calculations which utilize vectors and matrices. No diagram of the program flow for the MMATH subroutine was included herein either because, in large part, MMATH was duplicated from CSOFE.f. There were some additions made to MMATH for use in MMSOFE. In comparison to the original MMATH, the version of MMATH used in MMSOFE has the following added options:

1. New option to find the determinant of a square matrix of any size.
2. New option to transform a vector containing the upper triangular terms of a matrix into a symmetric matrix. While this operation and the next operation are performed in MSOFE via specific FORTRAN code, they are not performed in the MMATH subroutine in MSOFE.
3. New option to create a vector containing the upper triangular terms of a symmetric matrix.
4. Option to copy one column of an array to one column of another array, which can be used for building matrices from vectors or for pulling one column vector out of a matrix.

In order to clarify the MMAE code (included in Figure A.1 in Appendix A), which contains a number of calls to subroutine MMATH, a brief description of the MMATH call statement is given.

```
CALL MMATH( R,RR,CR, '=', A,RA,CA, '*', B,RB,CB )
```

The "R,RR,CR," "A,RA,CA," and "B,RB,CB," refer to the resultant array (R) and the two input arrays (A and B) with the number of rows (RR, RA, and RB) and number of columns (CR, CA, and CB) following the name of each array. The matrix functions possible with this version of MMATH are:

'Z'	MATRIX ZERO	$R = 0$
'='	MATRIX EQUIVALENCE	$R = A$
'+'	MATRIX ADDITION	$R = A + B$
'-'	MATRIX SUBTRACTION	$R = A - B$
'*'	MATRIX MULTIPLY	$R = A * B$
'X'	MATRIX MULTIPLY	$R = A \times B$
'F'	MATRIX FILL	$R = A(1,1)$
'T'	MATRIX TRANSPOSE	$R = A^T$
'S'	MATRIX SCALE	$R = A * B(1,1)$
'M'	MATRIX <- U-TRIANG VEC	$R(,) = A(,)$
'V'	U-TRIANG VEC <- MATRIX	$R(,) = A(,)$
'I'	COL R(I) <- COL A(J)	$R(,I) = A(,J)$
'D'	MATRIX DETERMINATE	$R = \langle A \rangle$
'I'	"VECTOR" INNER PRODUCT	$R = A^T * B$
'O'	"VECTOR" OUTER PRODUCT	$R = A * B^T$
'C'	VECTOR CROSS PRODUCT	$R = A \times B$
'N'	VECTOR NORM (MAGNITUDE)	$R = \langle A \rangle$
'U'	UNIT VECTOR	$R = A / \langle A \rangle$

Several of these need some added explanation.

- "MATRIX FILL" = A scalar value passed in a 1 X 1 matrix is loaded into all elements of matrix R.
- "COL R(I) <- COL A(J)" = The "J" column of matrix A is copied into the "I" column of matrix R. This can be used for stripping off one column vector from a matrix composed of column vectors.

3.4.4 Vectorization and Matricization. Once OPFILE was completed, the "vectorization" of the appropriate variables in MSOFE.f and USOFE.f was a major endeavor required before multiple-filter simulations of any kind could be done. The number of variables which were vectorized was kept to a minimum, although some variables were vectorized because it would be necessary for other applications, even though it might not have been necessary for this problem. The variables which needed to be vectorized were declared in either the 24 COMMON BLOCKS or in the subroutine-contained *local* declaration statements. These variables, as existing in MSOFE.f, are discussed in the MSOFE User's Manual (24) and are well described in MSOFE.f and USOFE.f programs themselves. Therefore, they will only be identified as having been vectorized, but their functions won't be described, except if vitally relevant.

Figure 3.8 shows the common blocks which were created for MMSOFE that did not exist in MSOFE. In Figure 3.8, the variable name is given first with its dimensions followed by its data type (IN = INTEGER, RL = REAL, DP = DOUBLE PRECISION, and etc.). Next is given a function descriptor such that

- "U" = Used in the subroutine.
- "S" = Set in the subroutine.
- "A" = Argument passed in the CALL statement.

COMMON BLOCK MMVAR contains almost all of the MMAE-related variables. While many of the variables in MMVAR could have been *locally* declared as *local* variables in subroutine MMAE, they were declared as COMMON variables in MMVAR to ease future adaptations utilizing the MMAE-related variables. With these variables in COMMON, a subroutine for another application could be added to USOFE.f and implemented merely by adding a call statement in MMSOFE, since all of the MMAE-related variables are available through the MMVAR COMMON. Further, no effort was made to economize on the number of variables, but instead a new variable was utilized for each new MMAE-related value calculated. This makes it easy to follow the calculations through the code, and it also makes it easier to write intermediate variables out to a file during development. The

MMVAR variables are described in Figure 3.9, where they are arranged in logical order, instead of alphabetically.

The COMMON BLOCK, DBUG, is vital to MMSOFE multiple model operation. It contains TTELF, which is used as the maximum number of files throughout MSOFE, ELF (elemental filter) which is the integer corresponding to the elemental filter currently being utilized and the xum_ vectors containing the FORTRAN unit numbers referred to in Table 3.2. The number suffixes on the xum_ vectors correspond to the hundreds digit of the MMSOFE unit number or the MSOFE unit number for the corresponding sets of files. If a "BL" file is accessed, either ELFT (ELF temporary) or TTELF are used, not ELF.

The vectorized variables and some new associated variables are specified in Figures 3.10 and 3.11 in accordance with where they are declared. In these figures and in MMSOFE, the variable, TELF, equals the total number of elemental filters, while the variable TTELF and the parameter DUMI both equal $TELF + 1$. The reason for having two is because DUMI is a parameter and is used to dimensionalize vectors in subroutine OPFILE; DUMI cannot be modified in the program, whereas TTELF is the true number of each type of file to open in subroutine OPFILE. Therefore, $TTELF = TELF + 1$ (the number of elemental filters plus one for the "BL" files) unless $TELF = 1$, in which case, $TTELF = 1$ because no "BL" files are opened. The MMSOFE filenames convention was discussed in Section 3.4. It is apparent from an inspection of Figure 3.11 that INIT (local initialization flag) was the only *local* variable vectorized in USOF.f, albeit in five USOF.f subroutines, and INIT was vectorized in most of the subroutines in MMSOFE.f. Figure 3.11 also lists all of the *local* variables in MMSOFE.f which were vectorized. Several special-use vectors were also declared as local variables for the simple purpose of using them as arguments in subroutine call statements. The only reason for the concern is that, if a call statement to a subroutine in MSOFE had a vector for an argument which was matricized in MMSOFE, the subroutine might not be passing the row or column desired. On some computers or with some FORTRAN compilers, matrices are stored in memory according to columns, while on other computers, the storage could be by rows. This method of using the special-use vectors obviated this argument ambiguity and eliminated any possibility of error in

passing between subroutines one row or one column of a matrix (in MMSOFE) which had formerly been a vector (in MSOFE). This practice also ensures the compatability of the code with other computer systems and FORTRAN compilers.

```

COMDECK MMVAR
COMMON/ MMVAR / XFMAT,HFILFM,ZFM,ZSM,RSM,RFM,PFMAT,XFM,
                PFMNS,XFMNS,HXFMNS,ZRESM,ZRESMT,HPF,
                HFILFMT, HPHTF,ALPHAM,ALPHAMI,ALPHAMIR,
                RTAR,DET,PROB, DPROB,DXF,PFCOL,PFM,PFMPLS,
                MINPROB,PFMM,DXFT,ZM,EM,DXFMAT,PFMMV,YY,
                PFMMVT,PFMMD,PAREST,PARMN,TRUPAR,EA,EP,
                EXPROB,EXCOEF,RTARV,PARCNT, MINFLAG
REAL XFMAT(NF,TELF), HFILFM(NZ,NF), ZFM(NZ,1),
     ZSM(NZ), RSM(NZ,NZ), RFM(NZ,NZ),
     PFMAT(NTF,TELF),XFM(NF), PFMNS(NF,NF),
     XFMNS(NF,1), HXFMNS(NZ,1), ZRESM(NZ,1),
     ZRESMT(1,NZ), HPF(NZ,NF), HFILFMT(NF,NZ),
     HPHTF(NZ,NZ), ALPHAM(NZ,NZ), ALPHAMI(NZ,NZ),
     ALPHAMIR(NZ,1), RTAR, DET(NZ,NZ),
     PROB(TELF,1), DPROB, DXF(NF),
     PFCOL(NTF), PFM(NF,NF), PFMPLS(NF,NF),
     MINPROB, PFMM(NF,NF), DXFT(1,NF),
     ZM(NZ), EM(NF), DXFMAT(NF,TELF),
     PFMMV(NTF), YY(NIN), PFMMVT(NTF),
     PFMMD(NF), PAREST(DUMI,PN),PARMN(PN),
     TRUPAR(PN), EA(PN), EP(PN),
     EXPROB(TELF,1), EXCOEF(TELF), RTARV(TELF)
INTEGER PARCNT, MINFLAG
COMDECK DBUG
COMMON/ DBUG / TTELF,ELF
                XUM02, XUM03, XUM04,XUM05, XUM06, XUM07,
                XUM08, XUM09, XUM10, XUM12, XUM15
INTEGER TTELF,ELF
                XUM02(DUMI), XUM03, XUM04(DUMI),XUM05(DUMI),
                XUM06(DUMI), XUM07(DUMI),XUM08(DUMI),
                XUM09, XUM10, XUM12, XUM15(DUMI)
COMDECK DBUG2
INTEGER DBG

```

Figure 3.8. NEW COMMON BLOCK Variables

ARGUMENTS

MOUT IN U A INDICATES ACTION TO TAKE
COMMON VARIABLES ACCESSED
/MMVAR/

XFMAT(NF,TELF)	RL() US	MATRIX OF XF - ALL ELEMENTAL FILTERS
HFILFM(NZ,NF)	RL() US	HFILF IN A 2D MATRIX
HFILFMT(NF,NZ)	RL() US	TRANPOSE OF HFILFM
ZFM(NZ,1)	RL() US	VECTOR OF FILTER MEASUREMENTS
ZSM(NZ)	RL() US	VECTOR OF SYSTEM MEASUREMENTS
RFM(NZ,NZ)	RL() US	VECTOR OF FILTER MEASUREMENT NOISE VARIANCES
RSM(NZ,NZ)	RL() US	VECTOR OF SYSTEM MEASUREMENT NOISE VARIANCES
PFMAT(NTF,TELF)	RL() US	MATRIX OF PF UPPER DIAG - ALL ELEMENTAL FILTERS
XFM(NF)	RL() US	MMAE BAYESIAN BLENDED STATE ESTIMATES
PFMNS(NF,NF)	RL() US	PF MINUS
XFMNS(NF,1)	RL() US	XF MINUS
HXFMNS(NZ,1)	RL() US	VECTOR OF FILTER MEASUREMENTS FOR EL RESIDUAL CALCULATION HXFMNS = HFILFM * XFMNS
ZRESM(NZ,1)	RL() US	EL FILTER RESIDUAL
ZRESMT(1,NZ)	RL() US	TRANPOSE OF ZRESM
HPF(NZ,NF)	RL() US	HFILFM * PFMNS
HPHTF(NZ,NZ)	RL() US	HFILFM * PFMNS * HFILFM(TRANPOSE)
ALPHAM(NZ,NZ)	RL() US	COVARIANCE OF MEASUREMENTS, $R+H*P*H(TRANPOSE)$
ALPHAMI(NZ,NZ)	RL() US	ALPHAM(INVERSE)
ALPHAMIR(NZ,1)	RL() US	ALPHAM(INVERSE) * RESIDUAL
RTAR	RL() US	.5 * RESIDUAL(TRANPOSE)*ALPHA(INVERSE)*RESIDUAL
DET(NZ,NZ)	RL() US	DETERMINANT OF THE ALPHAM
PROB(TELF,1)	RL() US	VECTOR OF CONDITIONAL PROBABILITY FOR EL FILTERS
DPROB	RL() US	DENOMINATOR OF CONDITIONAL PROBABILITY
DXF(NF)	RL() US	EL FILTERS - BLENDED STATE ESTIMATES (XF-XM)
DXFT(1,NF)	RL() US	DXF(TRANPOSE)
DXFMAT(NF,TELF)	RL() US	MATRIX OF DXF'S FOR ALL EL FILTERS
PFM(NF,NF)	RL() US	COVARIANCE MATRIX W/ DXF ($P=DXF*DXF(TRANPOSE)$)
PFCOL(NTF)	RL() US	COLUMN OF UPPER TRIANGLE OF PFM
MINPROB	RL() U	MINIMUM PROBABILITY BOUND (SET IN INPUT FILES)
PFMM(NF,NF)	RL() US	COVARIANCE OF BLENDED ESTIMATES
PFMMVT(NTF)	RL() US	COVARIANCE OF EACH ELEMENTAL FILTER $PFMMVT(I)=PROB(ELFT,1)*(PFMAT(I,ELFT)+PFCOL(I))$ $BLENDED COVARIANCE PFMMV=PROB(ELF)*PFMMV(ELF)$ $PFMMV(I)=PFMMV(I)+PFMMVT(I)$
PFMMV(NTF)	RL() US	
EM(NF)	RL() US	BLENDED ESTIMATION ERROR ($EM = XFM - AFS * XS$)
PAREST(DUMI,PN)	RL() US	BAYESIAN ESTIMATE OF THE PARAMETERS
TRUPAR(PN)	RL() US	TRUE VALUE OF THE PARAMETER FROM HRZSYS
EA(PN)	RL() US	ERROR IN THE PARAMETER ESTIMATE
EP(PN)	RL() US	DIAGONAL OF PARAMETER CALCULATED COVARIANCE
EXPROB(TELF,1)	RL() US	EL FILTER DENSITY ($EXCOEF * EXP(RTAR)$)
EXCOEF(TELF)	RL() US	LEADING COEFFICIENT OF DENSITY FOR EL FILTER
RTARV(TELF)	RL() S	VECTOR OF RTAR FOR ALL EL FILTERS, (OUTPUT ONLY)
MINFLAG	IN US	FLAG SET IF MINPROB IS USED
PN	IN U	NUMBER OF MEASUREMENTS
NZ	IN U	NUMBER OF MEASUREMENTS

Figure 3.9. NEW COMMON BLOCK Variables Descriptions

```

COMDECK CORBIT
  COMMON /CORBIT/ ZBOUND, ZFBias, RFNOIS,
    NWRf, NWRS, NWTF, NWTS, RNERR,
    RANGLF, RRANGF, RANGLS, RRANGS
  REAL TMPULS(2)MU,RHO,ZBOUND(TELF),ZFBias(TELF)
    RFNOIS(TELF),NWRf(TELF), NWRS(TELF),NWTF(TELF)
    NWTS(TELF),RANGLF(TELF),RRANGF(TELF),RANGLS,RRANGS
  INTEGER RNERR(TELF)
COMDECK SIZES
  INTEGER TELF,XUN15,DUMI,PN,NZ
  PARAMETER (PN = 2), (XUN15 = 101), (TELF = 5),(DUMI = TELF + 1),
    (TNF = TELF*NF), (TNS = TELF*NS),(NZ = 2)
COMDECK XUNITS
  INTEGER XUN02,XUN03,XUN04,XUN05,XUN06,XUN07,XUN08,XUN09,XUN10,
    XUN12,XUN15,XUM02(DUMI),XUM03,XUM04(DUMI),XUM05(DUMI),
    XUM06(DUMI),XUM07(DUMI),XUM08(DUMI),XUM09,XUM10,XUM12
  PARAMETER (XUN02=201, XUN03=3, XUN04=401, XUN05=501, XUN06=601)
    (XUN07=701, XUN08=801, XUN09=9, XUN10=10,XUN12=12)
COMDECK EVCNTL
  COMMON/ EVCNTL / DTFDBK, DTNOYS
  REAL DTFDBK(TELF), DTNOYS(TELF)
COMDECK ICBLK
  COMMON/ ICBLK / DV
  REAL DV(TELF,2)
COMDECK INCOND
  COMMON/ INCOND SEDXFI, SEDXSI
  INTEGER SEDXFI(TELF), SEDXSI(TELF)
COMDECK KOUNTS
  COMMON/ KOUNTS / NPP
  INTEGER NPP(TELF)
COMDECK MESDO
  COMMON/ MESDOA / ALPHA, ZF, ZRES, ZS, ZTOL
  REAL ALPHA(TELF),ZF(TELF),ZRES(TELF),ZS(TELF),ZTOL(TELF)
COMDECK MODFIL
  COMMON/ MODFIL / HF, HFILF, HFILS
  REAL HF(TELF,NF),HFILF(TELF,NF), HFILS(TELF,NS)
COMDECK MODSYS
  COMMON/ MODSYS / HS, HSYSF, HSYSS
  REAL HS(TELF,NS), HSYSF(TELF,NF), HSYSS(TELF,NS)
COMDECK MONOFF
  COMMON/ MONOFF / LCOV
  LOGICAL LCOV(TELF)

```

Figure 3.10. COMMON BLOCK variables

USOFE.f Local Variables

```
DECK DFDBAK
      INTEGER INIT(TELF)
DECK FCQFIL
      INTEGER INIT(TELF)
DECK FCQSYS
      INTEGER INIT(TELF)
DECK FSMAP
      INTEGER INIT(TELF)
DECK XFDBAK
      LOGICAL INIT(TELF)
```

MMSOFE.f Local Variables

```
DECK DERIV
      INTEGER INIT(TELF)
      REAL XSFDTI(TELF,NS+NF),DXSFDTI( NF+NS )
DECK EPRINT
      LOGICAL INIT(TELF)
DECK GETP
      INTEGER TELF
DECK KUTMER
      INTEGER INIT(TELF)
      REAL HDYN(TELF)
      LOGICAL ATSTEP(TELF) , ATTOUT(TELF)
DECK O7PLOT
      INTEGER INIT(TELF),ELFT
      INTEGER NXPTS(TELF)
DECK OUT
      INTEGER MACEPT(TELF) , MREJCT(TELF)
DECK RDGAIN
      INTEGER INIT(TELF)
DECK TNEXTE
      LOGICAL INIT(TELF)
DECK TNEXTM
      INTEGER INIT(TELF)
DECK VALDAT
      LOGICAL DATAOK(TELF)
DECK OPFILE
      INTEGER ELFT,DUMI,TTELF,I
      CHARACTER*7 FNAME(DUMI)
      CHARACTER*2 FNUMA(DUMI)
      CHARACTER*1 FNUM100(DUMI), FNUM10(DUMI)
      CHARACTER*1 FNUM1(DUMI)
      INTEGER FNUM(DUMI)
DECK MMAE
      REAL XSO(TELF,NS), XFO(TELF,NF)
      REAL PCO(TELF,NTC), PFO(TELF,NTF)
      REAL PEO(TELF,NTF), YTO(TELF,NYT)
      REAL EFO(TELF,NF), AFSo(TELF,NF,NS)
      REAL CFO(TELF,NF,NF), DFO(TELF,NF,NF)
      REAL FFO(TELF,NF,NF), QFO(TELF,NF,NF)
      REAL KFO(TELF,NF)
```

Figure 3.11. MMSOFE.f and USOFE.f Local Variables

The vectorized *local* variables in the two new subroutines, the OPFILE and MMAE subroutines, also require some explanation. In OPFILE, FNAME is the vector containing the entire file-names, including the elemental filter number or "BL" as derived by OPFILE. Similarly, the vector, FNUM, contains the FORTRAN unit number associated with a file. ELF100, ELF10, and ELF1 each correspond to the integer value of the hundreds, tens, and ones digit of the unit number, while the same representation in character form is given in variables FNUM100, FNUM10, and FNUM1. These are used to derive the prefix on the file names which correspond to the elemental filter number or "BL" for blended (see Table 3.2).

Because MMSOFE propagates and updates each elemental filter in turn before continuing to a new propagation/update cycle for any of the filters, vectors are required to save the current elemental-filter-specific values until the propagations and updates have been accomplished for all of the elemental filters. The saved values for an elemental filter are then reloaded when it is that filter's turn to propagate again. This permits the independent but parallel processing of the xx elemental filters. These *local* vectors are declared in the MMAE subroutine as shown in Figure 3.11. The MMSOFE-pertinent COMMON BLOCK variables are shown in Figure 3.10. Two variables whose values are supplied via the _INPUT files were shown in Figure 3.10 just to emphasize that they weren't vectorized. These vectors are:

1. TMPULSE = times to apply thrusts for Hohmann transfer.
2. RHO = radius ratio, apogee/perigee.

They weren't vectorized because these are part of the true (system) orbit determinations and the true orbit is the same for all elemental filters. They could be easily vectorized if required for some specific application. The other problem-specific (Group 5) variables from the _INPUT files which *were* vectorized are:

1. NWRF = filter range-rate process noise strength (note "R" for "range").
2. NWRS = system range-rate process noise strength.
3. NWTF = filter angle-rate process noise strength (note "T" for "theta").
4. NWTs = system angle-rate process noise strength.

5. RRANGF = filter range measurement noise variance.
6. RRANGS = system range measurement noise variance.
7. RANGLF = filter angle measurement noise variance.
8. RANGLS = system angle measurement noise variance.
9. ZBOUND = bound on acceptable measurement residual size, in multiple of filter-computed residual standard deviation, beyond which a single measurement will be edited out (no update performed).
10. MINPROB = lower bound on elemental filter probabilities.
11. RNERR = Number to specify the failure type to be simulated.
12. ZFBias = filter measurement bias.
13. RFNOIS = scale factor on filter measurement noise.

With some of these variables, vectorization may not have been strictly necessary, but it was implemented in order to simplify modifications later on. Essentially, if there was a question of whether to vectorize or not to vectorize at this time, it was chosen to "err" and vectorize, rather than be ultra-conservative and make future adaptations more difficult than necessary.

All of the "XUN0_" parameters in sizes and XUNITS provide the starting number for OPFILE to use in generating the input and output files. In the cases of XUN03, XUN09, XUN10, and XUN12, these are files correspond to one single MMSOFE file (see Table 3.2). All of the other vectorized variables given in Figure 3.10 required vectorization in order to keep the data and calculations separate for the elemental filters.

3.5 MMSOFE Validation

The MMSOFE validation corresponds to those software validation tests described in Section 1.6.4 as follows:

1. The failures chosen for MMSOFE validation (see Section 1.6.3 and Table 1.1) were programmed into subroutine HRZSYS to facilitate using the variable RNERR in the _INPUT files to designate the failure type.

2. The failure types shown in Table 1.1 were run with MSOFE to verify the tuning parameters and filter performance.
3. The MMSOFE software was partially validated using multiple, but identical elemental filters, without the implementation of MMAE probability and blending being computed. The MMSOFE-generated elemental filter data were compared to ensure all elemental filters were performing identically and then compared to data obtained from corresponding MSOFE runs. These data sets were compared "bit-for-bit." This comparison was done using the Matrix_x (10) software to difference a large number of pertinent variables and then plot those differences. An exact plot of zero for the differences indicated success. These numerous plots of straight lines at zero are not included in this thesis for obvious reasons. Suffice it merely to list the variables which were compared and checked. These variables were:

- XF = Filter state estimates
- XS = System (truth) state estimates
- EF = Filter/System state estimation error
- ZF = Filter (predicted) measurement
- ZS = System (actual) measurement
- PF = Vector of the upper triangular of the filter covariance matrix
- PC = Vector of the upper triangular of the composite covariance matrix
- PE = Vector of the upper triangular of the filter error covariance matrix
- ZRES = Measurement residual (actual - estimated)
- ALPHA = Estimated variance of the measurement residual which is calculated in MSOFE as a scalar iterative update rather than being calculated from a ZRES vector. This means there cannot be any "off diagonal" terms in this ALPHA.

4. The same values were again compared to validate the MMSOFE software by using multiple, but distinct elemental filters and comparing the elemental filter results to data obtained from matching MSOFE runs. Again the zero plots are not included.

5. Again the values were compared after the Bayesian-blending MMAE calculations were included. These comparisons were accomplished both with and without a Hohmann transfer in the profile.
6. The blended state estimates were compared to the state estimates obtained from the elemental filters and "sanity-checked" against plots of the residuals and probabilities generated by the elemental filters. Additionally, comparisons of the calculations were made to verify the accuracy of the Bayesian-blending code in the subroutine MMAE. Several passes through the multiple model MMAE calculations were compared with identical calculations done using the Matrixx (10) software. This comparison was done for calculations both before and after the time at which the failures were induced and repeated for several different failure types. No comparison plots were made of these comparisons, but the numerical values were verified visually on the computer screen to an accuracy of nine significant digits.

3.6 Summary

The goals of MMSOFE development have been to develop a tool for simulating multiple model adaptive estimation simulations which would be robust and easy to use for anyone familiar with MSOFE. Additionally and to test the new software, MMSOFE should be able to indicate failures via observing the Bayesian probabilities and also providing an estimate of the uncertain parameters. These goals have been met, and the MMSOFE tool developed should be as transportable to other computer systems as MSOFE.

IV. Analysis of Results

4.1 Overview

This chapter discusses the results of the MMSOFE multiple model simulations for the five failure types discussed in Section 1.6.3 and shown in Table 1.1, plus two simulations which were added to those in Table 1.1. Table 4.1 shows all seven failure types, including Cases #6 and #7, which were added to help clarify unforeseen MMAE-related phenomenon described in Section 4.3.2. Cases #6 and #7 and the rationale for adding them are described in Section 4.4.2. The plots which present the data from the simulations for each of the seven failure types are located in Appendix B, grouped into sections of plots corresponding to each failure type. Preceding each section of plots is also the related portion of Table 4.1. The explanations and variable definitions located with Table 1.1 in Section 1.6.3 also apply to Table 4.1. The plots are scaled uniformly according to what data is presented, with the plot scale chosen so as to include all of the data even in the worst data set. In other words, that all of the probability plots have a scale from 0 to 1 and the residual plots and the state estimation error plots all have a scale between + .5 and - .5. For Cases #1, #2, and #3, the parameter error plots are scaled from - .01 to + .01 and the parameter estimate plot range is from -.002 to .01. For Cases #4, #5, #6, and #7, the parameter error plots are scaled from - .2 to + .2 and the parameter estimate plot range is from -.03 to .15. By standardizing the plot scales as much as possible in this manner, it should be easier to make good comparisons between data presented on different plots. The plots corresponding to each failure type are organized as indicated in the introduction of Appendix B and also as follows:

1. Figure B.1: One plot of the probability associated with each elemental filter.
2. Figure B.2 and Figure B.3: Figure B.2 range and Figure B.3 for angle, each figure containing five separate plots. Each separate plot is related to an elemental filter and graphs the data for:
 - The mean residual.
 - Mean residual \pm Monte Carlo calculated standard deviations.

Table 4.1. Orbit Problem Failure Cases

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
1	Step in Measurement Noise	$R_S = R_{S0} + 6R_{S0}$	$T > 25$	$R_{f0}(1 + BIAS_{R_f})$
2	Ramp in Measurement Noise	$R_S = R_{S0} + \frac{6R_{S0}(T-25)}{(15)}$ $R_S = R_{S0} + 6R_{S0}$	$25 < T < 35$ $T > 35$	$R_{f0}(1 + BIAS_{R_f})$
3	Step/Return in Measurement Noise	$R_S = R_{S0}$ $R_S = R_{S0} + 6R_{S0}$	$T < 25, T > 35$ $25 < T < 35$	$R_{f0}(1 + BIAS_{R_f})$
4	Step in Measurement	$Z_S = Z_{S0} + .1Z_{S0}$	$T > 10$	$Z_f + BIAS_{Z_f}$
5	Ramp in Measurement	$Z_S = Z_{S0} + \frac{.1Z_{S0}(T-10)}{(10)}$ $Z_S = Z_{S0} + .1Z_{S0}$	$10 < T < 20$ $T > 20$	$Z_f + BIAS_{Z_f}$
6	Step in Measurement	$Z_S = Z_{S0} + .15Z_{S0}$	$T > 10$	$Z_f + BIAS_{Z_f}$
7	Ramp in Measurement	$Z_S = Z_{S0} + \frac{.15Z_{S0}(T-10)}{(15)}$ $Z_S = Z_{S0} + .15Z_{S0}$	$10 < T < 25$ $T > 25$	$Z_f + BIAS_{Z_f}$

- Zero mean \pm filter-predicted standard deviations.

3. Figures B.4 to B.7: Four figures, one corresponding to each of the four states (range, range-rate, theta, and theta-rate). Each figure consists of six separate plots. The plots correspond to the blended and five elemental filters and depict data for:

- The mean state estimation error.
- Mean estimation error \pm Monte Carlo calculated standard deviations.
- Zero mean \pm filter-predicted standard deviations.

4. Figure B.8: One figure with two plots relating to the MMAE/Bayesian parameter estimation:

- One plot of the mean estimation error, mean estimation error \pm Monte Carlo standard deviation, and zero \pm filter-predicted standard deviations.
- The second plot of the true parameter and mean estimated parameter values.

Plots occur sequentially thereafter in such groups of eight.

4.2 Elemental Filter Description

The five elemental filters (#1, #2, #3, #4, #5) were designed with Filter #3 always being the baseline filter -- no biases applied to the measurement nor to the measurement noise variance. For all of the simulations:

- Filter #1 has a measurement bias which matched the final value of the truth bias magnitude (except for Case #3 of Table 4.1).
- Filter #2 has a bias equal to $\frac{1}{2}$ of Filter #1's.
- For Cases #4 to #7 which have measurement bias/ramp-type failures, Filters #4 and #5 had measurement biases opposite in sign but equal in magnitude to those of Filters #2 and #1, respectively.
- For Cases #1 to #3 with the failure induced via a bias/ramp added to the measurement noise variance, negative values for the measurement noise variance would not be correct, so the Filter #4 variance bias halved the baseline noise variance magnitude and Filter #5 halved Filter #4's variance bias magnitude. Referring to Table 4.1, this halving was accomplished by setting $BIAS_{R_i} = -.5$ in Case #4 and $BIAS_{R_i} = -.75$ in Case #5, which corresponds to the labels on Figures B.1 through B.23.

The particular discretization of the parameter space utilized in this thesis made it easier to observe probability results but it is not necessarily the best discretization for state or parameter estimates. Refinement of the parameter discretization was not a consideration in this thesis effort.

4.3 Probabilities

4.3.1 Expected Probabilities. Because Filter model #3 *matches* the truth model in the absence of failures, one might expect its probability to be highest when there is no failure. Additionally, Filter #1 *matches* the true situation when the induced failure has reached its maximum (full-failure) and Filter #2 is midway between Filters #3 and #1 and therefore Filter #2 *matches* the situation midway between no-failure and full-failure (important particularly in ramp-type failure modes). Heuristically, it might be expected

that the probabilities in Case #2 (ramping of the measurement noise variance) should shift from Filter #3, pass to Filter #2 as the failure magnitude grows, and then shift over to Filter #1 when the failure is fully culminated. In Cases #1 and #3, it might be expected that the probabilities would switch directly from Filter #3 to Filter #1 with only a small increase (if any) in Filter #2's probability. These expectations were proven to be essentially valid for Cases #1, #2, and #3 in which the failure was induced in the measurement noise variance (see Figures B.1, B.9, B.17), but not for the other cases in which the failure was induced directly via the measurement (see Figures B.25 and B.33). Filter #2's probability did increase after failure induction for Cases #1 to #3, but Filter #2 never became dominant.

4.3.2 Unexpected Probability Phenomena. The heuristics broke down somewhat for the cases in which the failure was induced as a "weak" ramp directly on the measurements (Cases #5, See Table 4.1), growing to a bias equal in magnitude (.1 distance units) to the measurement failure bias (Cases #4). They also broke down for Case #4 in which the bias was introduced suddenly as a "small" step (Appendix B, Section B.4). Both the "small" step and the "weak" ramp cases (Appendix B, Section B.5) culminated with a bias of .1 distance units in the measurement. In both of these cases, the probabilities did not shift all the way over to Filter #1 but stayed "hung-up" on the intermediate Filter #2. At least they shifted away from the baseline filter (#3), but sticking on Filter #2 was not the expected result.

This phenomenon can be understood, though, by examining the residuals which are the sources of the information for the hypothesis conditional probability calculations within the MMAE and by keeping in mind that a probability determined at one measurement update time carries over into the probability calculation performed at the next update time by direct weighting of the density function (Equation (2.27)). Therefore, once Filter #2 dominated the probability, its residuals would have to become very "bad" and another filter's residuals would have to become "good" to shift probability from Filter #2 to that other filter. Recognition of this unexpected phenomenon is made more important because of the relationship of these cases to spoofing. All four cases, #4, #5, #6, and #7 (Section

4.4.2), represent the simulation of spoofing which was discussed in Chapter 1 and by Vasquez (36) with regard to a Global Positioning System. This phenomenon indicates how difficult detection and isolation of gradually growing spoofing-type errors really are, even when using MMAE techniques.

4.4 Analysis of Simulation Data

4.4.1 *Failure Induced by Measurement Noise Variance Bias or Ramp.* Cases #1, #2, and #3 (see Figures B.1 to B.24) all represent failures akin to signal jamming in the Global Positioning System, as discussed in Chapter 1 and by Vasquez (36). This is a simpler failure type to detect than the spoofing type failures of Cases #4 through #7. Consequently, there were no great surprises, as those which will be described in Section 4.4.2. Case #1 represents the total loss of GPS signal or signal jamming after $T = 25$. Case #2 represents a gradual fading out of the signal or gradual application of a jamming signal from $T = 25$ to $T = 35$ and then maintaining of the level at the end of the ramp for the remainder of the simulation. Case #3 represents the same conditions as Case #1 but with the jamming or signal loss ending at $T = 35$.

Initially, during the benign (no-failure) part of all three of these simulations, the probabilities for Filters #2, #4 and baseline all vied for the highest probability. It seems heuristically obvious that these three filters should have the highest probabilities because these three filters most closely match the truth in the no-failure situation. By about $T = 12$, the baseline elemental filter had become dominant with the highest probability and remained dominant. The fact that the baseline filter ultimately remained dominant in the absence of any failure was confirmed by one simulation with $T_{final} = 100$ (the plots of which are not included in this thesis).

In Cases #1 and #3 which had the failure induced via a measurement noise variance bias beginning at $T = 25$, the probability began shifting from the baseline filter to Filter #1 immediately after inception of the failure. With Case #3, the probability began shifting back from Filter #1 to the baseline filter when the failure was discontinued at $T = 35$. The probability shifted much more slowly from the baseline filter to Filter #1 in Case #2 because the ramped failure caused a much slower change in the residuals and therefore a

slower shifting of the probability. Figure B.10 is key to understanding why the probability mass leaves Filter #3 and is attracted to Filter #1 vs. sequentially to Filter #2 and then to Filter #1. By the time Filter #3's residuals are poor enough to cause p_3 to decrease, Filter #1's residuals are better (relative to the filter-computed Λ_1) than those of Filter #2. Plot B.16 shows good identification of Filter #2, though it is very similar to Plot B.8 in which the parameter estimate would be anticipated to converge more quickly to a_1 .

It is also interesting to note the following similarities in the probabilities calculated for Cases #1, #2, and #3.

1. The probabilities for Cases #1, #2, and #3 were identical (as they should be) for all five filters until $T = 25$ when the failures were induced.
2. They were identical until $T = 35$, for Cases #1 and #3, at which time the failure in Case #3 was turned off (see Figures B.1 and B.17).
3. Likewise, the probabilities for the five filters were almost the same for Cases #1 and #2 by the end of the simulation at $T = 50$ because the failure offset had been the same for these two Cases since $T = 35$.
4. Presumably, the probabilities for Cases #1 and #2 would have become equal sometime after $T = 50$.

4.4.2 Failure Induced by Measurement Bias or Ramp. Referring to Table 4.1 and to the figures in Appendix A, which depict the residuals from the measurement ramp and bias failure cases, the following observations are of interest:

1. The angle residual plots (Figures B.27, B.35, B.43 and B.51) provided no clear insights.
2. As anticipated, the baseline filter had the best range residuals and the highest probabilities before the failure onset in all four of these cases (#4, #5, #6, and #7).
3. For Cases #4 and #5 (.1 bias and ramp cases):
 - Filter #2 attracted and kept the highest probability from the baseline filter right after the failure was induced and while Filter #1, which intuitively should have

acquired the highest probability, responded slightly in probability accumulation but then lost out to Filter #2.

- Examination of the corresponding range residuals (Figures B.26 and B.34) reveals that the Filter #2 range residual (mean residual values over the 15 Monte Carlo runs) was within one filter-predicted standard deviation of zero throughout the entire simulation. The mean range residual for Filter #1 was outside one filter-predicted standard deviation of zero before failure induction even though it was essentially zero after the full failure realization. It was hypothesized that the probability could be forced toward the "correct" filter either by allowing a longer simulation time over which Filter #1's residuals would have better characteristics than those in Filter #2, or by letting the truth model maximum offset be larger so as to make Filter #2's residuals more distinctly poor relative to Filter #1's residuals. The hypothesis of allowing the longer simulation time was tried without the desired result. This then led to Cases #6 and #7.

4. For Cases #6 and #7 (.15 bias and ramp cases):

- While Filter #2 began to attract the probability from the baseline filter just as in Cases #4 and #5, Filter #1 soon took over and had the highest probability for the rest of the simulation, just as originally hypothesized.
- Examining the corresponding range residuals clarifies the situation. While the residuals are identical to those of Cases #4 and #5 before induction of the failure, the larger magnitude of the final bias value caused the range residuals of Filter #1 to be worse than they were in Cases #4 and #5, but caused those for Filter #2 to become *much* worse. In fact, the range residuals for Filter #2 are slightly more than one standard deviation from zero and those for Filter #1 somewhat less. Thus, the probability can switch from the baseline filter first to Filter #2 and finally over to Filter #1, as originally anticipated.

These observations support the postulate presented in Section 4.3.2. The range residuals from Cases #4 and #5 were better for Filter #2, and the Filter #2 probability higher than those for Filter #1, immediately after the failure was introduced. Therefore, Filter

#2 continued to attain higher and higher probabilities according to Equation (2.26). By the time Filter #1's residuals became better than Filter #2's, Filter #2 had good residuals and higher probabilities, so Filter #1 couldn't attract the probability over the time of the simulation. However, with Cases #6 and #7, Filter #2's range residuals became so much worse than those of Filter #2, that Filter #2 gave up the probability to Filter #1, which had better range residuals.

The observations with regard to Cases #4 and #5 were the impetus for the last two test cases which hadn't originally been planned but were retro-fitted into Table 1.1 as the test Cases #6 and #7 in Table 4.1. Case #6 has a truth measurement failure bias of .15 distance units in contrast to .1 distance units of Case #4 (see Appendix B, Section B.6 and Figures B.41 - B.48). Case #7 is the case in which the failure was induced as a "strong" ramp on the measurement (see Appendix B, Section B.7 and Figures B.49 - B.56). The failure was induced identically to the prior "weak" ramp test case (Case #5), with the ramps of equal slope but with ramp endurance of $1\frac{1}{2}$ times as long for Case #5. Therefore, for Case #7, the failure bias on the measurement was .15 instead of just .1 distance units after the slope ended at $T = 25$.

4.4.3 State Estimation. The MMAE-blended state estimates for all four states appear quite accurate from the state estimation error plots shown in Appendix B. In each of these cases, the blended estimation error mean value, standard deviation of that error, and the filter-predicted standard deviation were smaller than they were for any of the five elemental filters. While not accomplished in this research, a good performance baseline could be created by running a single filter simulation in which the filter is artificially informed of the exact nature and time of failure.

The accuracy of the state estimates for Cases #4 - #7 was excellent before the failures were induced, with mean estimation errors of nearly zero and small standard deviations of the estimation error. After the failures were induced, the blended state estimation error was as small as the state estimation error corresponding to the elemental filter with the highest probability. Thus, for Cases #4 and #5, the errors were as small as Filter #2's (highest probability) even though Filter #1 had smaller state estimation errors, essentially

zero mean. After introduction of the failure, the standard deviations of the estimation errors followed a pattern similar to that of the state estimation error mean values which could be expected because they were probability dependent also.

4.4.4 Parameter Estimation. Before the failures were induced in the simulations, the parameter estimates were nearly exact for all of the cases simulated. However, after the failure was induced, because Filter #1 in each of the first five error cases modelled the final magnitude of the induced failure exactly, the parameter estimates obtained in all of these five cases (Table 4.1) must be smaller than the value hypothesized in Filter #1. Additionally, the parameter estimate could only be that large if the probability for Filter #1 were equal to 1, which is not possible since the other probabilities were lower bounded at .01. This upper bounding of the parameter estimate is due to the fact that the MMAE-based parameter estimate is the sum of the probability-weighted parameters hypothesized in each elemental filter. If the estimated parameter were equal to or greater than the true value, the probabilities would also have to sum to something greater than one. That is not a possible solution, and thus the parameter estimates *must* be biased in this application.

The parameter estimation errors for Cases #1 - #3 were generally smaller in the steady state after the failure was fully realized than for Cases #4 - #7. For Cases #4 - #7, the steady state parameter estimation error mean values were all nearly equal (-.05). For Cases #6 and #7 in which the hypothesized parameter value in Filter #1 was .1 and the true parameter value was .15, worse parameter estimates might have been anticipated. However, except for the transient period during and slightly after the failure was applied, the parameter estimation error was nearly the same as for Cases #4 and #5.

Failure Detection and Isolation (FDI) with MMAE (see Sections 1.5.4 and 2.3) is accomplished merely by monitoring the probabilities for each of the elemental filters or, equivalently, by monitoring the parameter estimate. For the test cases (see Table 4.1) used in this thesis, the elemental filter which modelled the correct failure was detected for all cases except Cases #4 and #5. In these two cases, Filter #2 was identified instead of correctly identifying Filter #1. This difficulty could possibly have been overcome by

additional discretization studies in which more elemental filters, some with hypothesized parameters exceeding the possible true parameter values. That might have resulted in correct FDI and improved the state and parameter estimates as well. Another possibility to overcome the difficulty is to restart the probabilities periodically at the lower bound value for all elemental filters except the baseline filter (perhaps in parallel with the original MMAE probabilities). In this scenario, as probability "dumps out of" Filter #3 when its residuals are large, the probability mass ought to be absorbed by the filter with the best-looking residuals.

4.5 Summary.

Chapter IV has described the results of the MMSOFE orbit problem simulations. Analysis of the elemental filter performance, the blended state estimates, the probabilities and parameter estimation, and the FDI results were discussed.

V. Conclusions and Recommendations

This chapter provides a brief discussion of conclusions reached with regard to the MMSOFE software and the simulation results. Recommendations for future research are also presented.

5.1 Conclusions

5.1.1 MMSOFE. The primary goal of this thesis (see Section 1.2) was to develop a simulation and analysis software tool to aid controls designers and system integrators in developing and analyzing MMAE applications. As described in Chapter III, the input and output file organization used by MSOFE (24) was preserved. As discussed in detail in Chapter IV, MMSOFE performed the Kalman filter calculations for each elemental filter of the MMAE, which was verified by bit-for-bit data comparisons with data produced using the same filter models in MSOFE. MMSOFE also calculates the probabilities associated with the correctness of each elemental filter, calculates the MMAE blended state estimates and MMAE-computed state estimation error covariance, and estimated the parameter values. MMSOFE will process from one to ninety-eight elemental filters merely by modifying the file SIZES and by generating an input file for each elemental filter.

Inherent to the primary goal were several other goals:

- The MSOFE structure was modified as little as possible in developing MMSOFE.
- The MMSOFE software tool should be user-friendly for anyone already familiar with MSOFE
- The method of implementing MMAE via MMSOFE could make it easy to adapt MMSOFE for other multiple model applications as well. The method of using the subroutine SIMRUN as the MMSOFE top-level-executive routine and to handle the coordination of which elemental filter is being processed makes the MMSOFE adaptation for other applications quite simple. Thus MMSOFE could be easily expanded to accommodate applications such as MMAE-based control, multiple model adaptive control, or distributed Kalman filtering.

5.1.2 Simulations. The detailed discussion of the results from the satellite orbit estimation problem was presented in Chapter IV and will not be repeated. Suffice it to say that:

- The performance of MMSOFE's calculations for each elemental filter was verified as accurate based on comparison to MSOFE.
- The MMAE-based calculation of probabilities, state estimates, parameter estimates, and FDI for the cases simulated were entirely reasonable and were deemed accurate.

5.2 Recommendations

A brief list of recommendations for future research is presented, with many of the details concerning these items presented earlier.

- Adapt MMSOFE for MMAE-based control.
- Adapt MMSOFE for multiple model adaptive control.
- Adapt MMSOFE for distributed Kalman filtering and compare the results with the results from other DKF software tools, such as Distributed Kalman Filter Simulation (DKFSIM) (3).
- Use MMSOFE to apply MMAE to the integrated navigation reference system investigated by Vasquez in his thesis (36). The navigation reference system integrates an inertial navigation system, a range/range-rate transponder system, and a Global Positioning System.
- Use MMSOFE for other applications, such as those listed in Section 1.5.3.
- Since MMSOFE estimates the parameters using probability weighting, MMSOFE could be used as an aid to tune a filter optimally. This could be accomplished by distributing the discretized parameter space across the elemental filters as with any MMAE simulation, letting the uncertain parameters be the filter tuning parameters. The estimate of the parameter could then be used for subsequent MMSOFE or MSOFE simulations.
- The previous recommendation could be changed slightly and MMSOFE modified to take partial derivatives of variables of interest with respect to the discretized

parameter values. These variables of interest could be state estimates, covariances, or estimation errors (or their statistics). These partial derivatives could then be utilized by a gradient algorithm to help find optimal tuning parameters or, for that matter, to help find optimal values for uncertain parameters located in any part of the filter model.

Appendix A. New MMSOFE Subroutines

This appendix contains the MMSOFE FORTRAN code for the two completely new subroutines, MMAE and OPFILE.

A.1 MMAE Subroutine

```
*DECK MMAE
      SUBROUTINE MMAE ( MOUT )
C
C1-----
C      AVIONICS DIRECTORATE                      MMSOFE      1.5
C      WRIGHT LABORATORY                        JUNE      1991
C      WRIGHT-PATTERSON AFB OHIO                COPYRIGHT
C-----
C      NOTICE: THIS ROUTINE IS SUBJECT TO THE DISTRIBUTION LIMITATION,
C      EXPORT WARNING, AND DESTRUCTION NOTICE AT TOP OF FILE.
C-----
C      OUT      --  OUTPUT CONTROLLER
C
C      USAGE    --  CALL MMAE ( MOUT )
C
C      ABSTRACT --  MMAE CONTROLS THE CALCULATIONS RELATED TO
C                   MULTIPLE MODEL CALCULATIONS. THE TABLE BELOW GIVES
C                   MORE DETAIL ABOUT WHAT HAPPENS IN THE INDIVIDUAL OUTPUT
C                   ROUTINES FOR EACH VALUE OF MOUT.
C
C
C      LIMITATIONS --  NONE KNOWN
C
C      REMARKS  --
C                   TABLE OF OUTPUT ACTIONS(**) FOR EACH POSSIBLE EVENT
C-----
C      MOUT  EVENT
C      INDEX DESCRIPTOR
C-----
C      1  INITIALIZE PROBLEM
C      2  INITIALIZE RUN
C      3  COPY VECTORS TO WORKING VECTORS FROM LAST ITERATION
C      4  SAVE WORKING VECTORS FOR NEXT ITERATION
C      5  WRITE MMAE DATA TO OUTPUT FILE BLCNTOM
C      6  SAVE XF FOR MMAE STATE ESTIMATES & SAVE PF INTO MATRIX FORM
C          FOR CALCULATION OF REAL ALFA
C      7  PERFORM MMAE CALCULATION REQUIRED FOR EACH ELEMENTAL FILTER
C      8  PERFORM CALCULATIONS FOR MULTIPLE MODEL ADAPTIVE ESTIMATION
C          ONLY WHEN THE LAST ELEMENTAL FILTER HAS BEEN COMPLETE FOR
C          THIS PROPAGATION/UPDATE CYCLE
C-----
C      HEADER DATE  --  93/10/29
```

```

C ARGUMENTS
C MOUT IN U A INDICATES ACTION TO TAKE
C
C COMMON VARIABLES ACCESSED
C /MMVAR/
C XPMAT(NF,TELF) RL() US MATRIX OF XF - ALL ELEMENTAL FILTERS
C HFILFM(NZ,NF) RL() US HFILF IN A 2D MATRIX
C HFILFMT(NF,NZ) RL() US TRANSPOSE OF HFILFM
C ZFM(NZ,1) RL() US VECTOR OF FILTER MEASUREMENTS
C ZSM(NZ) RL() US VECTOR OF SYSTEM MEASUREMENTS
C RFM(NZ,NZ) RL() US VECTOR OF FILTER MEASUREMENT NOISE VARIANCES
C RSM(NZ,NZ) RL() US VECTOR OF SYSTEM MEASUREMENT NOISE VARIANCES
C PFMAT(NF,TELF) RL() US MATRIX OF PF UPPER DIAG - ALL ELEMENTAL FILTERS
C XPM(NF) RL() US MMAE BAYESIAN BLENDED STATE ESTIMATES
C PFMNS(NF,NF) RL() US PF MINUS
C XPMNS(NF,1) RL() US XF MINUS
C HXPMNS(NZ,1) RL() US VECTOR OF FILTER MEASUREMENTS FOR EL
C RESIDUAL CALCULATION HXPMNS = HFILFM * XPMNS
C ZRESM(NZ,1) RL() US EL FILTER RESIDUAL
C ZRESMT(1,NZ) RL() US TRANSPOSE OF ZRESM
C HPF(NZ,NF) RL() US HFILFM * PFMNS
C HPHTF(NZ,NZ) RL() US HFILFM * PFMNS * HFILFM(TRANPOSE)
C ALPHAM(NZ,NZ) RL() US COVARIANCE OF MEASUREMENTS, R+H*P*H(TRANPOSE)
C ALPHAMI(NZ,NZ) RL() US ALPHAM(INVERSE)
C ALPHAMIR(NZ,1) RL() US ALPHAM(INVERSE) * RESIDUAL
C RTAR RL() US .5 * RESIDUAL(TRANPOSE)*ALPHA(INVERSE)*RESIDUAL
C DET(NZ,NZ) RL() US DETERMINANT OF THE ALPHAM
C PROB(TELF,1) RL() US VECTOR OF CONDITIONAL PROBABILITY FOR EL FILTERS
C DPROB RL() US DENOMINATOR OF CONDITIONAL PROBABILITY
C DXF(NF) RL() US EL FILTERS - BLENDED STATE ESTIMATES (XF-XM)
C DXFT(1,NF) RL() US DXF(TRANPOSE)
C DXPMAT(NF,TELF) RL() US MATRIX OF DXF'S FOR ALL EL FILTERS
C PFM(NF,NF) RL() US COVARIANCE MATRIX W/ DXF (P=DXF*DXF(TRANPOSE))
C PFCOL(NF) RL() US COLUMN OF UPPER TRIANGLE OF PFM
C MINPROB RL() U MINIMUM PROBABILITY BOUND (SET IN INPUT FILES)
C PFMH(NF,NF) RL() US COVARIANCE OF BLENDED ESTIMATES
C PFMHVT(NF) RL() US COVARIANCE OF EACH ELEMENTAL FILTER
C PFMHVT(I)=PROB(ELFT,1)*(PFMAT(I,ELFT)+PFCOL(I))
C PFMHV(NF) RL() US BLENDED COVARIANCE PFMHV=PROB(ELF)*PFMHV(ELF)
C PFMHV(I)=PFMHV(I)+PFMHVT(I)
C EM(NF) RL() US BLENDED ESTIMATION ERROR (EM = XFM - AFS * XS)
C PAREST(DUMI,PW) RL() US BAYESIAN ESTIMATE OF THE PARAMETERS
C TRUPAR(PW) RL() US
C EA(PW) RL() US
C EP(PW) RL() US
C EXPROB(TELF,1) RL() US EL FILTER DENSITY (EXCOEF * EXP(RTAR))
C EXCOEF(TELF) RL() US LEADING COEFFICIENT OF DENSITY FOR EL FILTER
C RTARV(TELF) RL() S VECTOR OF RTAR FOR ALL EL FILTERS, (OUTPUT ONLY)
C MINFLAG IN US FLAG SET IF MINPROB IS USED
C /RUNTIM/
C NRUNS IN U NUMBER OF RUNS IN SIMULATION STUDY
C /USRDM/
C PC RL() U A COMPOSITE COVARIANCE VECTOR, UPPER TRIANGLE
C PF RL() U A FILTER COVARIANCE VECTOR, UPPER TRIANGLE
C T DP U A SIMULATION TIME
C XF RL() U A FILTER STATE VECTOR

```

```

C XS      RL()  U A   SYSTEM TRUTH STATE VECTOR
C /USPAR/
C IRUN    IN    U     CURRENT RUN NUMBER, 1 TO NRUNS
C
C LOCAL VARIABLES
C
C PARAMETERS
C NF      IN    U     NUMBER OF STATES IN FILTER MODEL
C NS      IN    U     NUMBER OF STATES IN SYSTEM TRUTH MODEL
C NTC     IN    U     SIZE OF COMPOSITE COVARIANCE TRIANGLE, NC(NC+1)/2
C NTF     IN    U     SIZE OF FILTER COVARIANCE TRIANGLE, NF(NF+1)/2
C NYT     IN    U     NUMBER OF VARIABLES IN TRAJECTORY VECTOR
C TELF    IN    U     NUMBER OF ELEMENTAL FILTERS
C PN      IN    U     NUMBER OF ELEMENTAL FILTERS
C TELF    IN    U     NUMBER OF ELEMENTAL FILTERS
C DOMI    IN    U     NUMBER OF ELEMENTAL FILTERS + 1
C TTSELF  IN    U     NUMBER OF ELEMENTAL FILTERS + 1
C PN      IN    U     NUMBER OF MEASUREMENTS
C
C ROUTINES USED
C EMFIL    INITPR    INITRM    INVERT    MMATH
C2-----
      IMPLICIT CHARACTER (A-Z)
C *** PARAMETERS
      INCLUDE 'SIZES.'
C *** ARGUMENTS
      INTEGER MOUT
C *** COMMON VARIABLES
      INCLUDE 'XUNITS.'
      INCLUDE 'DEBUG.'
      INCLUDE 'DEBUG2.'
      INCLUDE 'USPAR.'
      INCLUDE 'ICBLK.'
      INCLUDE 'MMVAR.'
C *** COMMON VARIABLES
      INCLUDE 'CONSTS.'
      INCLUDE 'EVCNTL.'
      INCLUDE 'MONOFF.'
      INCLUDE 'USRDM.'
      INCLUDE 'RUNTIM.'
C *** COMMON VARIABLES
      INCLUDE 'MESDO.'
      INCLUDE 'MESIM.'
      INCLUDE 'UDMAT.'
      INCLUDE 'MODSYS.'
      INCLUDE 'MODFIL.'
C *** LOCAL VARIABLES
      INTEGER I,J,K,ELFT
      REAL    XSO(TELF,NS),    XFO(TELF,NF),
&            KFO(TELF,NF),
&            PCO(TELF,NTC),    PFO(TELF,NTF),
&            PEO(TELF,NTF),    YTO(TELF,NYT),
&            EFO(TELF,NF),    AFPO(TELF,NF,NS),
&            CFO(TELF,NF,NF), DFO(TELF,NF,NF),
&            FFO(TELF,NF,NF), QFO(TELF,NF,NF)
C

```

```

C3-----
C4-----
C
  IF ( MOUT .EQ. 1 ) THEN
C
C  #INITIALIZE PROBLEM
C
  IF (DBG.GE.2) WRITE (XUM15(ELF),
& ('*****CALL INITPR*****ELF= ',I4))ELF
  IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE
& (0,('*****CALL INITPR*****ELF= ',I4))ELF
    CALL INITPR
C
  ELSEIF ( MOUT .EQ. 2 ) THEN
C
C  #INITIALIZE RUN
C
    IF (T.LE..00001) THEN
      PARCNT=0
      DO 5 J=1,PN
        PARMN(J)=0.0
5      CONTINUE
    ENDIF
C
    IF (DBG.GE.2) WRITE (XUM15(ELF),
& ('*****CALL INITRN***ELF,IRUN= ',2I4))ELF,IRUN
    IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE
& (0,('*****CALL INITRN***ELF,IRUN= ',2I4))ELF,IRUN
    IF (DBG.GE.2) WRITE(XUM06(ELF),
& ('*****CALL INITRN***ELF,IRUN= ',2I4))ELF,IRUN
      CALL INITRN
C
    DO 35 ELFT=1,TELF
      PROB(ELFT,1) = REAL(1.0/TELF)
35    CONTINUE
C
  ELSEIF ( MOUT .EQ. 3 ) THEN
C
C  #COPY VECTORS TO WORKING VECTORS FROM LAST ITERATION
    DO 101 I=1,NS
      XS(I)=XSO(ELF,I)
101    CONTINUE
    DO 102 I=1,NF
      XF(I)=XFO(ELF,I)
      EF(I)=EFO(ELF,I)
      KF(I)=KFO(ELF,I)
      DO 103 J=1,NF
        CF(I,J)=CFO(ELF,I,J)
        DF(I,J)=DFO(ELF,I,J)
        FF(I,J)=FFO(ELF,I,J)
        QF(I,J)=QFO(ELF,I,J)
103      CONTINUE
      DO 104 J=1,NS
        AFS(I,J)=AFSO(ELF,I,J)
104      CONTINUE
102    CONTINUE

```

```

DO 105 I=1,NTC
  PC(I)=PCO(ELF,I)
105 CONTINUE
DO 106 I=1,NTF
  PF(I)=PFO(ELF,I)
  PE(I)=PEO(ELF,I)
106 CONTINUE
DO 107 I=1,NYT
  YT(I)=YTO(ELF,I)
107 CONTINUE
C
  ELSEIF ( MOUT .EQ. 4 ) THEN
C    #SAVE WORKING VECTORS FOR NEXT ITERATION
    DO 111 I=1,NS
      XSO(ELF,I)=XS(I)
111 CONTINUE
    DO 112 I=1,NF
      XFO(ELF,I)=XF(I)
      EFO(ELF,I)=EF(I)
      KFO(ELF,I)=KF(I)
      DO 113 J=1,NF
        CFO(ELF,I,J)=CF(I,J)
        DFO(ELF,I,J)=DF(I,J)
        FFO(ELF,I,J)=FF(I,J)
        QFO(ELF,I,J)=QF(I,J)
113 CONTINUE
      DO 114 J=1,NS
        AFSO(ELF,I,J)=AFS(I,J)
114 CONTINUE
112 CONTINUE
    DO 115 I=1,NTC
      PCO(ELF,I)=PC(I)
115 CONTINUE
    DO 117 I=1,NTF
      PFO(ELF,I)=PF(I)
      PEO(ELF,I)=PE(I)
117 CONTINUE
    DO 119 I=1,NYT
      YTO(ELF,I)=YT(I)
119 CONTINUE
C
  ELSEIF ( MOUT .EQ. 5 ) THEN
C    #WRITE MMAE DATA TO OUTPUT FILE BLCNTOM
    WRITE(XUM02(TTELF))
    & REAL(T),
    & ((XFMAT(I,J),I=1,NF),J=1,TELF),
    & XFM,
    & PFMM,
    & PFMMD,
    & EM,
    & ZM,
    & PROB,
    & (PAREST(DUMI,J),J=1,PW),
    & EA,
    & EP,
    & TRUPAR

```

```

C
ELSEIF ( MOUT .EQ. 6 ) THEN
C
C      $SAVE XF FOR MMAE STATE ESTIMATES & SAVE PF INTO MATRIX FORM
C      FOR CALCULATION OF REAL ALFA
C
C      CALL MMATH( PFMNS,NF,NF, 'M', PF,NTF,1, 'M', PF,NTF,1 )
C
C      CALL MMATH( XFMNS,NF,1, '=' , XF,NF,1, '=' , XF,NF,1 )
C
ELSEIF ( MOUT .EQ. 7 ) THEN
C
C      $PERFORM MMAE CALCULATION REQUIRED FOR EACH ELEMENTAL FILTER
C
C      DO 12 I=1,NF
C          XFMAT(I,ELF)=XF(I)
12      CONTINUE
C      DO 13 I=1,NTF
C          PFMAT(I,ELF)=PF(I)
13      CONTINUE
C      $
C      $CALCULATE ZRES() = ZF() - H(.) * XF()
C      $CALCULATE (R' * A * R)/2
C      CALL MMATH( HXFMNS,NM,1, '=' , HFILFM,NM,NF, '*' , XFMNS,NF,1 )
C
C      CALL MMATH( ZRESM,NM,1, '=' , ZSM,NM,1, '~', HXFMNS,NM,1 )
C      $
C      $CALCULATE ALPHA(.) = H(.) * P(.) * H(.) + R()
C      CALL MMATH( HPP,NM,NF, '=' , HFILFM,NM,NF, '*' , PFMNS,NF,NF)
C
C      CALL MMATH(HFILFMT,NF,NM, '=' , HFILFM,NM,NF, 'T', HFILFM,NM,NF)
C
C      CALL MMATH(HPHTF,NM,NM, '=' , HPF,NM,NF, '*' , HFILFMT,NF,NM)
C
C      CALL MMATH( ALPHAM,NM,NM, '=' , RFM,NM,NM, '+', HPHTF,NM,NM)
C
C      $
C      $CALCULATE ALPHAMI(.) = ALPHAM(.)
C
C      CALL INVERT ( ALPHAM, NM, ALPHAMI )
C
C      $
C      $CALCULATE TRANSPOSE OF ZRESMT() = ZRESM()
C
C      CALL MMATH( ZRESMT,1,NM, '=' , ZRESM,NM,1, 'T', ZRESM,NM,1 )
C
C      $CALCULATE EXPONENTIAL FOR DENSITY
C      $
C      $ R() * A * R()
C
C      CALL MMATH(ALPHAMIR,NM,1, '=' , ALPHAMI,NM,NM, '*' , ZRESM,NM,1)
C
C      CALL MMATH(RTAR,1,1, '=' , ZRESMT,1,NM, '*' , ALPHAMIR,NM,1 )
C
C      RTAR = -RTAR/2.0
C      RTARV(ELF) = RTAR

```

```

C      #CALCULATE SQRT OF DETERMINANT OF ALPHAM,
C      #CALCULATE COEFFICIENT (SCALE FACTOR PRECEDING EXPONENTIAL)
C      #CALCULATE THE DENSITY = COEFFICIENT * EXPONENTIAL
C
C      CALL MMATH(DET,NM,NM,'=',ALPHAM,NM,NM,'D',ALPHAM,NM,NM)
C
C      EXCOEF(ELF)= 1.0/(((2.0*3.14159236)**(NM/2))*SQRT(DET(1,1)))
C      EXPROB(ELF,1)=EXP(RTAR)*EXCOEF(ELF)
C
C      ELSEIF ( MOUT .EQ. 8 ) THEN
C
C      #PERFORM CALCULATIONS FOR MULTIPLE MODEL ADAPTIVE ESTIMATION
C      ONLY WHEN THE LAST ELEMENTAL FILTER HAS BEEN COMPLETE FOR
C      THIS PROPAGATION/UPDATE CYCLE
C
C      #CALCULATE THE CONDITIONAL PROBABILITIES FOR EACH ELEMENTAL FILTER
C      # 1ST CALCULATE THE DENOMINATOR (= SUM OF THE DENSITIES)
C
C      MINFLAG=0
C      DPROB=0.0
C      DO 135 ELFT=1,TELF
C
C          DPROB=DPROB+PROB(ELFT,1)*EXPROB(ELFT,1)
135      CONTINUE
C
C      #CALCULATE THE CONDITIONAL PROBABILITIES
C
C      DO 140 ELFT=1,TELF
C          PROB(ELFT,1)=PROB(ELFT,1)*EXPROB(ELFT,1)/DPROB
C          IF (PROB(ELFT,1).LT.MINPROB) THEN
C              MINFLAG=3
C              PROB(ELFT,1)=MINPROB
C          ENDIF
140      CONTINUE
C      IF (MINFLAG.GT.1) THEN
C          DPROB=0.0
C          DO 145 ELFT=1,TELF
C              DPROB=DPROB+PROB(ELFT,1)
145          CONTINUE
C          DO 148 ELFT=1,TELF
C              PROB(ELFT,1)=PROB(ELFT,1)/DPROB
148          CONTINUE
C          MINFLAG=0
C      ENDIF
C
C      #CALCULATE THE BLENDED STATE ESTIMATES (XFM)
C
C      CALL MMATH( XFM,NF,1,'=',XFMAT,NF,TELF,'*',PROB,TELF,1 )
C
C      #CALCULATE THE VARIANCE OF STATES FROM EACH ELEMENTAL FILTER
C      FROM THE BLENDED OPTIMAL STATE ESTIMATES & TRANSPOSE
C      DO 150 ELFT=1,TELF
C          DO 355 I=1,NF
C              DXF(I)=XFMAT(I,ELFT)-XFM(I)
C              DXFMAT(I,ELFT)=DXF(I)
355      CONTINUE

```

```

      CALL MMATH( DXFT,1,NF, '=', DXF,NF,1, 'T', DXF,NF,1 )
C
C      *CALCULATE COVARIANCE OF THE ELEMENTAL FILTERS wrt. THE BLENDED
C      T
C      ESTIMATES, P = DXF * DXF
C
      DO 176 I=1,NF
        DO 177 K=1,NF
          PFM(I,K)=DXF(I)*DXFT(1,K)
177      CONTINUE
176      CONTINUE
C
C      *TURN THAT COVARIANCE MATRIX INTO A VECTOR
C
      CALL MMATH(PFCOL,NTF,1,'V',PFM,NF,NF,'V',PFM,NF,NF )
C
C      *CALC BLENDED COVARIANCE PFMM = PROB(ELF)*PFMM(ELF) MATRIX
C
      DO 180 I=1,NTF
        PFMHVT(I)=PROB(ELFT,1)*(PFMAT(I,ELFT)+PFCOL(I))
        PFMHV(I)=PFMMV(I)+PFMHVT(I)
180      CONTINUE
C
C      *STRIP OFF JUST ONE COLUMN OF ELEMENTAL FILTER U-D COVARIANCES
C
C      *CALC BLENDED COVARIANCE PFMM = PROB(ELF)*PFMM(ELF) MATRIX
150      CONTINUE
      CALL MMATH(PFMM,NF,NF,'=',PFMHV,NTF,1,'M',PFMHV,NTF,1 )
C
C      *COMPUTE BLENDED FILTER ERROR EM IF REQUIRED:
C
      CALL EMFIL
C
C      *CALCULATE THE ESTIMATES OF THE PARAMETER
C
      DO 202 J=1,PN
        PAREST(DUMI,J)=0.0
        DO 201 ELFT=1,TELF
          PAREST(DUMI,J)=PAREST(DUMI,J)+PAREST(ELFT,J)*PROB(ELFT,1)
201      CONTINUE
          PARMN(J)=((PARMN(J)*PARCNT)+PAREST(DUMI,J))/(PARCNT+1)
          EA(J)=PAREST(DUMI,J)-TRUPAR(J)
          EP(J)=(PAREST(DUMI,J)-TRUPAR(J))**2
          PARCNT=PARCNT+1
202      CONTINUE
        DO 190 I=1,NTF
          PFMHV(I)=0.0
190      CONTINUE
C
      ENDIF
C
C      *RETURN TO CALLING PROGRAM UNIT.
C
      RETURN
      END

```

A.2 OPFILE Subroutine

```

*DECK OPFILE
      SUBROUTINE OPFILE(XUN,GENAME,FRM,RCL,STATS)
C
C1=====
C      AVIONICS DIRECTORATE                MSOFE      1.5
C      WRIGHT LABORATORY                  JUNE      1991
C      WRIGHT-PATTERSON AFB OHIO          COPYRIGHT
C=====
C      NOTICE: THIS ROUTINE IS SUBJECT TO THE DISTRIBUTION LIMITATION,
C      EXPORT WARNING, AND DESTRUCTION NOTICE AT TOP OF FILE.
C=====
C      OPFILE  --  OUTPUT FILE NAMER & NUMBERER
C
C      USAGE  --  CALL  OPFILE(XUN,GENAME,FRM,RCL,STATS)
C
C      ABSTRACT --  OPFILE ASSIGNS NUMBERS AND NAMES TO THE OUTPUT FILES:
C      XUN02 , CNTOM --> XUN02(ELFT) , CNTOM(ELFT)      *
C      XUN04 , DSTOM --> XUN04(ELFT) , DSTOM(ELFT)
C      XUN05 , MSOFE_IN --> XUN05(ELFT) , INPUT(ELFT)    *
C      XUN06 , MDATA --> XUN06(ELFT) , MDATA(ELFT)
C      XUN07 , Scratch --> XUN07(ELFT)
C      XUN08 , ERRS --> XUN08(ELFT) , ERRRS(ELFT)
C      XUN03 , FLIGHT --> ??? future ???
C      XUN09 , OLDEND --> ??? future ???
C      XUN10 , NEWEND --> ??? future ???
C      XUN12 , KFGAIN --> ??? future ???
C      * WHERE 'ELFT' IS THE ELEMENTAL FILTER NUMBER,
C      *      s.t. 0 < ELFT <= TELF + 1.
C      * WHERE TELF IS THE NUMBER OF ELEMENTAL FILTERS
C      * AND WHERE THE FILE NAMES BEGIN WITH THE VALUE OF ELFT.
C
C
C      LIMITATIONS  --  NONE KNOWN
C
C      HEADER DATE  --  93/02/22
C
C      ARGUMENTS
C      XUN02  IN   U      UNIT NUMBER FOR FILE "01CNTOM"
C      XUN03  IN   U      UNIT NUMBER FOR FILE "FLIGHT"
C      XUN04  IN   U      UNIT NUMBER FOR FILE "01DSTOM"
C      XUN05  IN   U      UNIT NUMBER FOR FILE "01INPUT"
C      XUN06  IN   U      UNIT NUMBER FOR FILE "01MDATA"
C      XUN07  IN   U      UNIT NUMBER FOR FILE "01" SCRATCH FILE
C                          FOR ACCUMULATING PLOT DATA
C      XUN08  IN   U      UNIT NUMBER FOR FILE "01ERRRS"
C      XUN09  IN   U      UNIT NUMBER FOR FILE "OLEND"
C      XUN10  IN   U      UNIT NUMBER FOR FILE "NUEND"
C      XUN12  IN   U      UNIT NUMBER FOR FILE "KFGAIN"
C      TELF   IN   U      NUMBER OF ELEMENTAL FILTERS
C      DUMI   IN   US     TELF + 1
C

```

C COMMON VARIABLES ACCESSED

C /DEBUG/

C TTSELF IN US EQUAL TO DUMI
 C ELF IN U NUMBER OF ELEMENTAL FILTER BEING PROCESSED
 C XUM02 IN() U UNIT NUMBER FOR FILE "__CNTOM"
 C XUM03 IN U UNIT NUMBER FOR FILE "FLIGHT"
 C XUM04 IN() U UNIT NUMBER FOR FILE "__DSTOM"
 C XUM05 IN() U UNIT NUMBER FOR FILE "__INPUT"
 C XUM06 IN() U UNIT NUMBER FOR FILE "__MDATA"
 C XUM07 IN() U UNIT NUMBER FOR FILE "__" SCRATCH FILE
 C FOR ACCUMULATING PLOT DATA
 C XUM08 IN() U UNIT NUMBER FOR FILE "__ERRRS"
 C XUM09 IN() U UNIT NUMBER FOR FILE "OLEND"
 C XUM10 IN() U UNIT NUMBER FOR FILE "WUEND"
 C XUM12 IN() U UNIT NUMBER FOR FILE "KFGAIN"

C

C LOCAL VARIABLES

C

C XUN IN U VALUE PASSED TO OPFILE FOR STARTING FILE NUMBER
 C ELFT IN U TEMPORARY VARIABLE OF ELEMENTAL FILTER NUMBER
 C I IN U LOOPING VARIABLE
 C FRM IN U "FORMAT" OF THE FILE
 C RCL IN U
 C FNAME IN U
 C STATS IN U
 C GENAME IN U
 C FNUMA IN U
 C FNUM100 IN U
 C FNUM10 IN U
 C FNUM1 IN U
 C FNUM IN U
 C ELF100 IN U
 C ELF10 IN U
 C ELF1 IN U

C

C ROUTINES USED NONE

C2

C

IMPLICIT CHARACTER (A-Z)

C *** PARAMETERS

INCLUDE 'SIZES.'
 INCLUDE 'XUNITS.'

C *** ARGUMENTS

C *** COMMON VARIABLES

CDEB *** COMMON VARIABLES

INCLUDE 'DEBUG.'
 INCLUDE 'DEBUG2.'
 INCLUDE 'USRDMN.'

C *** LOCAL VARIABLES

INTEGER ELFT,DUMI,TTSELF,I
 CHARACTER*11 FRM
 CHARACTER*7 FNAME(DUMI)
 CHARACTER*7 STATS
 CHARACTER*5 GENAME
 CHARACTER*2 FNUMA(DUMI)
 CHARACTER*1 FNUM100(DUMI),FNUM10(DUMI),FNUM1(DUMI)

```

INTEGER FNUM(DUMI),XUM,RCL
INTEGER*2 ELF100,ELF10,ELF1

```

C

C3-----

C4-----

```

      TTSELF = DUMI
      IF (TSELF .EQ. 1) TTSELF = 1
      DO 99 ELFT = 1,TTSELF
        FNUM(ELFT)=ELFT+XUM-1
        IF (ELFT .NE. TTSELF) THEN
100      ELF100=FNUM(ELFT)/100
          ELF10=(FNUM(ELFT)-(ELF100*100))/10
          ELF1=FNUM(ELFT)-(ELF100*100)-(ELF10*10)
          FNUM100(ELFT)=CHAR(ELF100+48)
          IF (FNUM(ELFT) .LT. 100) FNUM100(ELFT)='0'
          FNUM10(ELFT)=CHAR(ELF10+48)
          IF (FNUM(ELFT) .LT. 10) FNUM10(ELFT)='0'
          FNUM1(ELFT)=CHAR(ELF1+48)
          FNUMA(ELFT)=FNUM10(ELFT)//FNUM1(ELFT)
          IF (DBG.GE.2) WRITE (XUM15(ELF),5994)
          & FNUMA(ELFT),FNUM100(ELFT),FNUM10(ELFT),
          & FNUM1(ELFT),GENAME
          IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE (0,5994)
          & FNUMA(ELFT),FNUM100(ELFT),FNUM10(ELFT),
          & FNUM1(ELFT),GENAME
        ELSE
          FNUMA(ELFT)='BL'
          IF (TTSELF .EQ. 1) FNUMA(ELFT)=''
          FNUM100(ELFT)=''
          FNUM10(ELFT)=''
          FNUM1(ELFT)=''
          IF (DBG.GE.2) WRITE (XUM15(ELF),5995)
          & FNUMA(ELFT),FNUM100(ELFT),FNUM10(ELFT),
          & FNUM1(ELFT),GENAME
          IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE (0,5995)
          & FNUMA(ELFT),FNUM100(ELFT),FNUM10(ELFT),
          & FNUM1(ELFT),GENAME
        ENDIF
        FNAME(ELFT)= FNUMA(ELFT)//GENAME
        IF ((FNUM(ELFT).GT.100).AND.(FNUM(ELFT).LE.200))
          & XUM15(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.200).AND.(FNUM(ELFT).LE.300))
          & XUM02(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.400).AND.(FNUM(ELFT).LE.500))
          & XUM04(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.500).AND.(FNUM(ELFT).LE.600))
          & XUM05(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.600).AND.(FNUM(ELFT).LE.700))
          & XUM06(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.700).AND.(FNUM(ELFT).LE.800))
          & XUM07(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.800).AND.(FNUM(ELFT).LE.900))
          & XUM08(ELFT) = FNUM(ELFT)
        IF ((FNUM(ELFT).GT.12).AND.(FNUM(ELFT).LE.20))
          & XUM15(ELFT) = FNUM(ELFT)

```

```

      IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE (0,5993)
&      ELFT,FNUM(ELFT),FNAME(ELFT),FRM,RCL,STATS
      IF (TTSELF.EQ. 1) FNAME(ELFT)=GENAME
      IF ((XUM .GE. 500).AND.(XUM .LT.600)) THEN
        IF (DBG.GE.2) WRITE (XUM15(ELF),(''OPEN500''))
        IF ((DBG.GT.0).AND.(DBG.LE.2))
&        WRITE(0,(''OPEN500''))
        OPEN ( UNIT = FNUM(ELFT),
&          FILE = FNAME(ELFT),
&          FORM = FRM,
&          STATUS = STATS )
      ELSEIF ((XUM .GE. 700).AND.(XUM .LT.800)) THEN
        IF (DBG.GE.2) WRITE (XUM15(ELF),(''OPEN700''))
        IF ((DBG.GT.0).AND.(DBG.LE.2))
&        WRITE(0,(''OPEN700''))
        OPEN ( UNIT = FNUM(ELFT),
&          FORM = 'UNFORMATTED',
&          STATUS = 'SCRATCH' )
      ELSE
        IF (DBG.GE.2)
&        WRITE(XUM15(ELF),(''OPEN???'))
        IF ((DBG.GT.0).AND.(DBG.LE.2))
&        WRITE(0,(''OPEN???'))
        OPEN ( UNIT = FNUM(ELFT),
&          FILE = FNAME(ELFT),
&          FORM = FRM,
&          RECL = RCL,
&          STATUS = STATS )
      ENDIF
    ENDIF

```

C

```

      IF ((DBG.GT.0).AND.(DBG.LE.2)) THEN
        WRITE(0,5997) IRUN,ELFT,GENAME,FNUM(ELFT),FNAME(ELFT),T
        WRITE(0,5990)(XUM02(I),I=1,DUMI)
        WRITE(0,5991)(XUM04(I),I=1,DUMI)
        WRITE(0,5987)(XUM05(I),I=1,DUMI)
        WRITE(0,5992)(XUM06(I),I=1,DUMI)
        WRITE(0,5989)(XUM07(I),I=1,DUMI)
        WRITE(0,5988)(XUM08(I),I=1,DUMI)
        WRITE(0,5977)(XUM15(I),I=1,DUMI)
      ENDIF
99  CONTINUE
5990 FORMAT ('XUM02 = ',3(I4))
5991 FORMAT ('XUM04 = ',3(I4))
5987 FORMAT ('XUM05 = ',3(I4))
5992 FORMAT ('XUM06 = ',3(I4))
5989 FORMAT ('XUM07 = ',3(I4))
5988 FORMAT ('XUM08 = ',3(I4))
5977 FORMAT ('XUM15 = ',3(I4))
5993 FORMAT ('5993 ',I4,' FNUM = ',I4,' FNAME = ',A,
&          ' FRM = ',A,' RCL = ',I4,' STATS = ',A)
5994 FORMAT ('5994',4FNUMA(ELFT)=',A',FNUM100(ELFT) ',
&          ',A,' //FNUM10(ELFT) ',A,' //FNUM1(ELFT) ',A,'GENAME',A)
5995 FORMAT ('5995',5FNUMA(ELFT)=',A',FNUM100(ELFT) ',
&          ',A,' //FNUM10(ELFT) ',A,' //FNUM1(ELFT) ',A,'GENAME',A)
5996 FORMAT ('5996',FNUM(' ,I4,' ) = ',I4)

```

```

5997 FORMAT ('5997 IRUM = ',I4,' ELFT = ',I4,' GENAME = ',A,
& ' FNAME(',I4,') = ',A,' T = ',F7.3)
5998 FORMAT ('5998','OPFILE ',A,' ',I4,' ',A,' ',I4)
5999 FORMAT ('5999','OPFILE IRUM T TI '
& 'TF TIN TOUT TUP TMAX TNEXT TSAVE TMEAS ',A)
6000 FORMAT ('6000',' ELFT and TELF = (',I4,' ',I4,')')
6001 FORMAT ('6001',' TTELF and DUMI = (',I4,' ',I4,')')
IF (DBG.GE.2) WRITE (XUM15(ELF),
& (''RETURN FROM OPFILE *****''))
IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE (0,'
& ('' RETURN FROM OPFILE *****''))
IF (DBG.GE.2)
& WRITE (XUM15(ELF),('' RETURN FROM OPFILE''))
IF ((DBG.GT.0).AND.(DBG.LE.2)) WRITE (0,('' '''))
IF (DBG.GE.2) WRITE (XUM15(ELF),('' '''))
RETURN
END

```

Appendix B. *Simulation Plots*

This appendix contains the plots of the multiple model orbit estimation problem simulations, including plots for each failure type as discussed in Section 4.1 and shown in Table 4.1. For each of these failures, the Figures contain the following plots (Figure references are given for the first failure type only):

1. Figure B.1: One plot of the probability associated with each elemental filter.
2. Figure B.2 and Figure B.3: Two figures, one for range and one for angle, each figure containing five separate plots. Each separate plot is related to an elemental filter and graphs the data for:
 - The mean residual.
 - Mean residual \pm Monte Carlo calculated standard deviations.
 - Zero mean \pm filter-predicted standard deviations.
3. Figures B.4 to B.7: Four figures, one corresponding to each of the four states. Each figure consists of six separate plots. The plots correspond to the blended and five elemental filters and depict data for:
 - The mean state estimation error.
 - Mean estimation error \pm Monte Carlo calculated standard deviations.
 - Zero mean \pm filter-predicted standard deviations.
4. Figure B.8: One figure with two plots relating to the MMAE/Bayesian parameter estimation:
 - One plot of the mean estimation error, mean estimation error \pm Monte Carlo standard deviation, and zero \pm filter-predicted standard deviations.
 - The second plot of the true parameter and mean estimated parameter values.

Plots occur sequentially thereafter in such groups of eight.

B.1 Measurement Noise Bias at $T > 25$

Table B.1. Orbit Problem Failure Case #1

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
1	Step in Measurement Noise	$R_S = R_{S0} + 6R_{S0}$	$T > 25$	$R_{f0}(1 + BIAS_{R_f})$

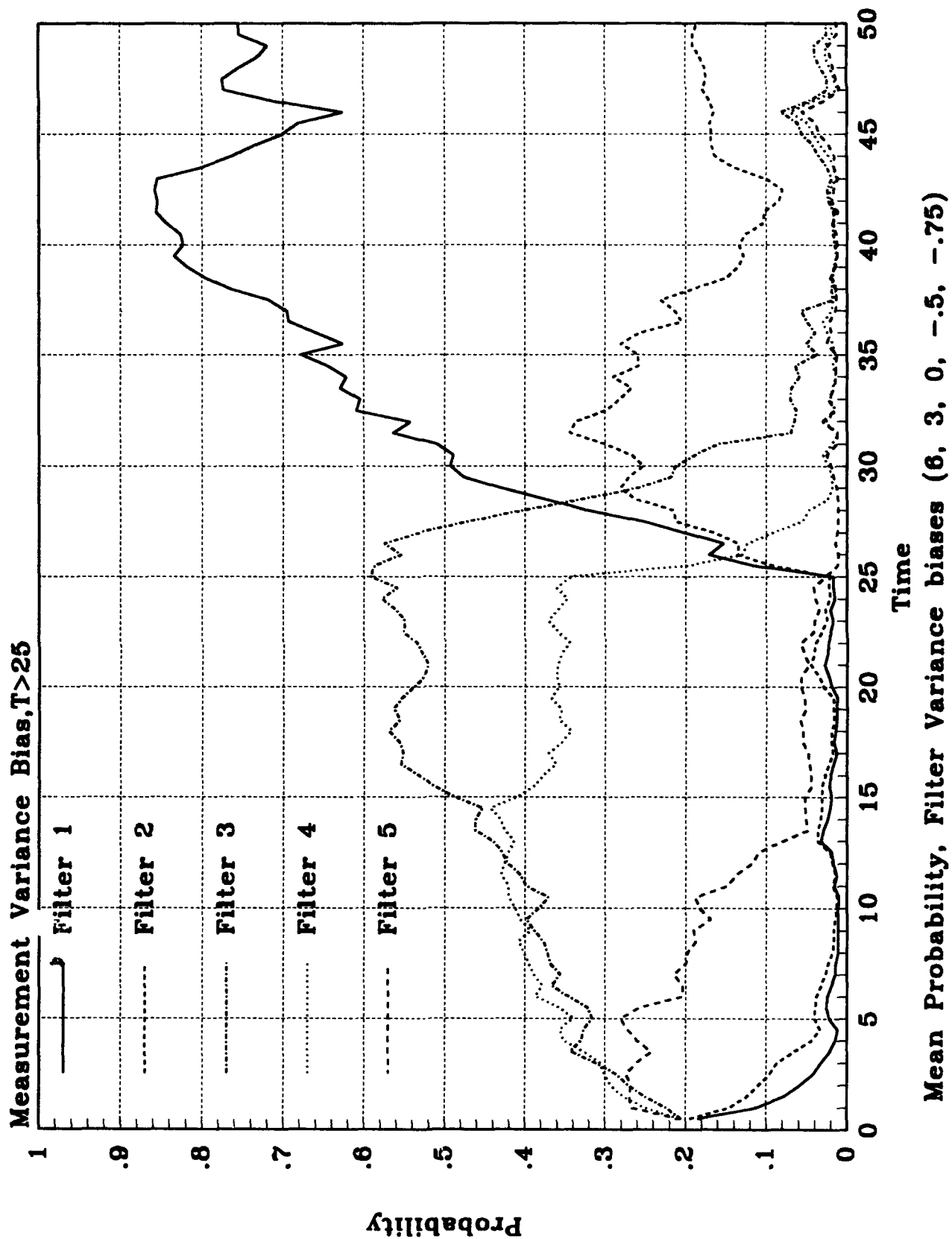


Figure B.1. Case #1 Probabilities

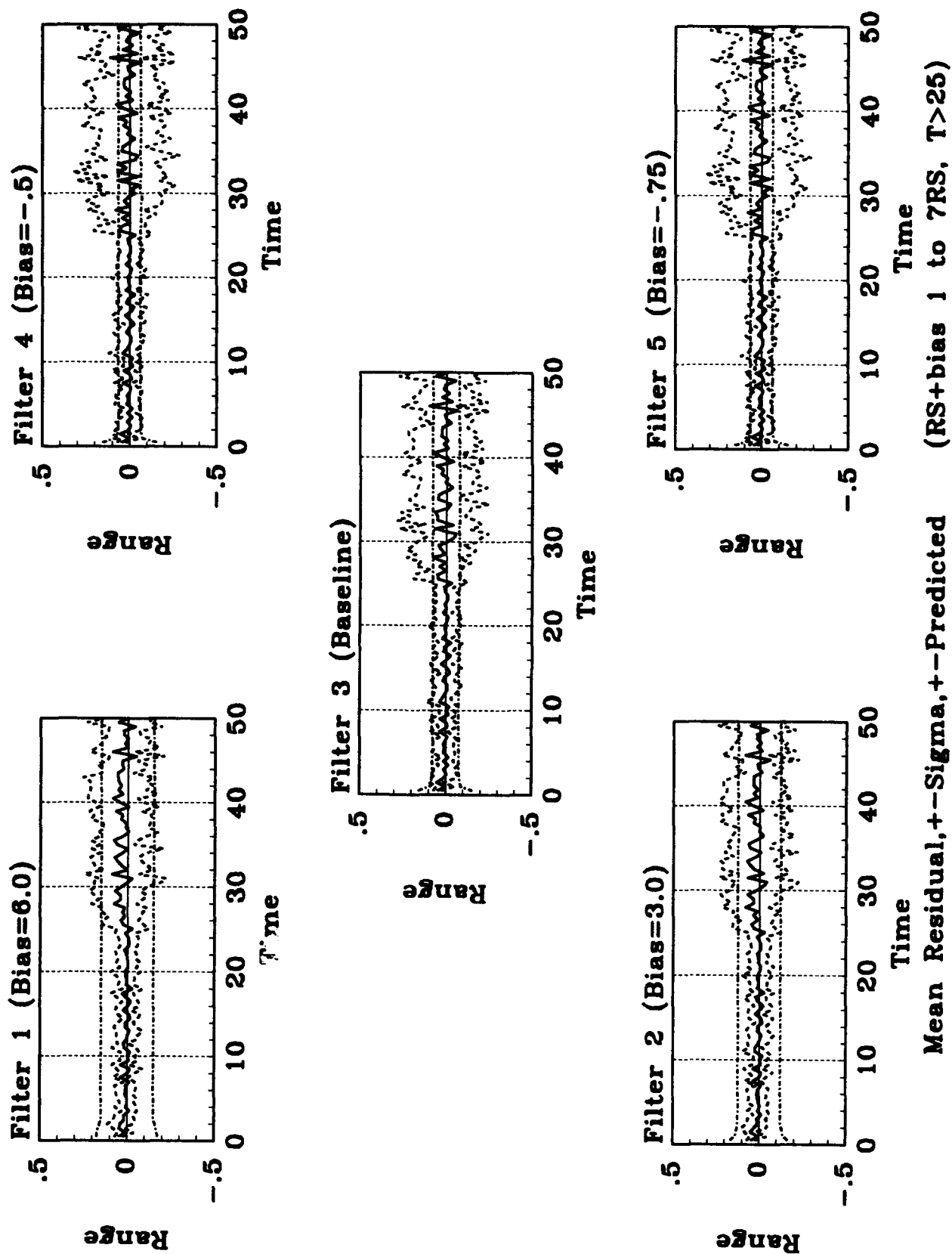


Figure B.2. Case #1 Range Residuals
B-4

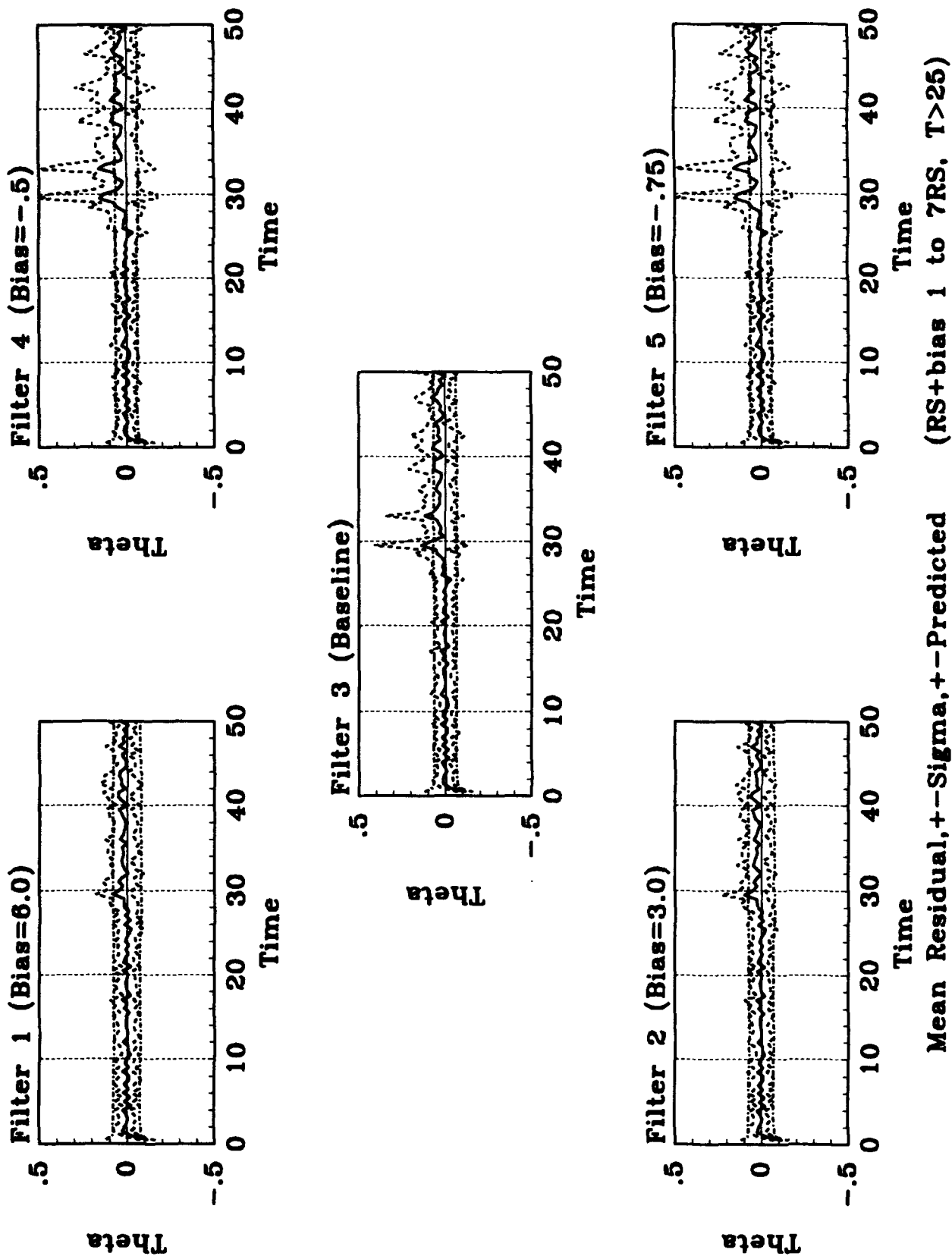


Figure B.3. Case #1 Theta Residuals
B-5

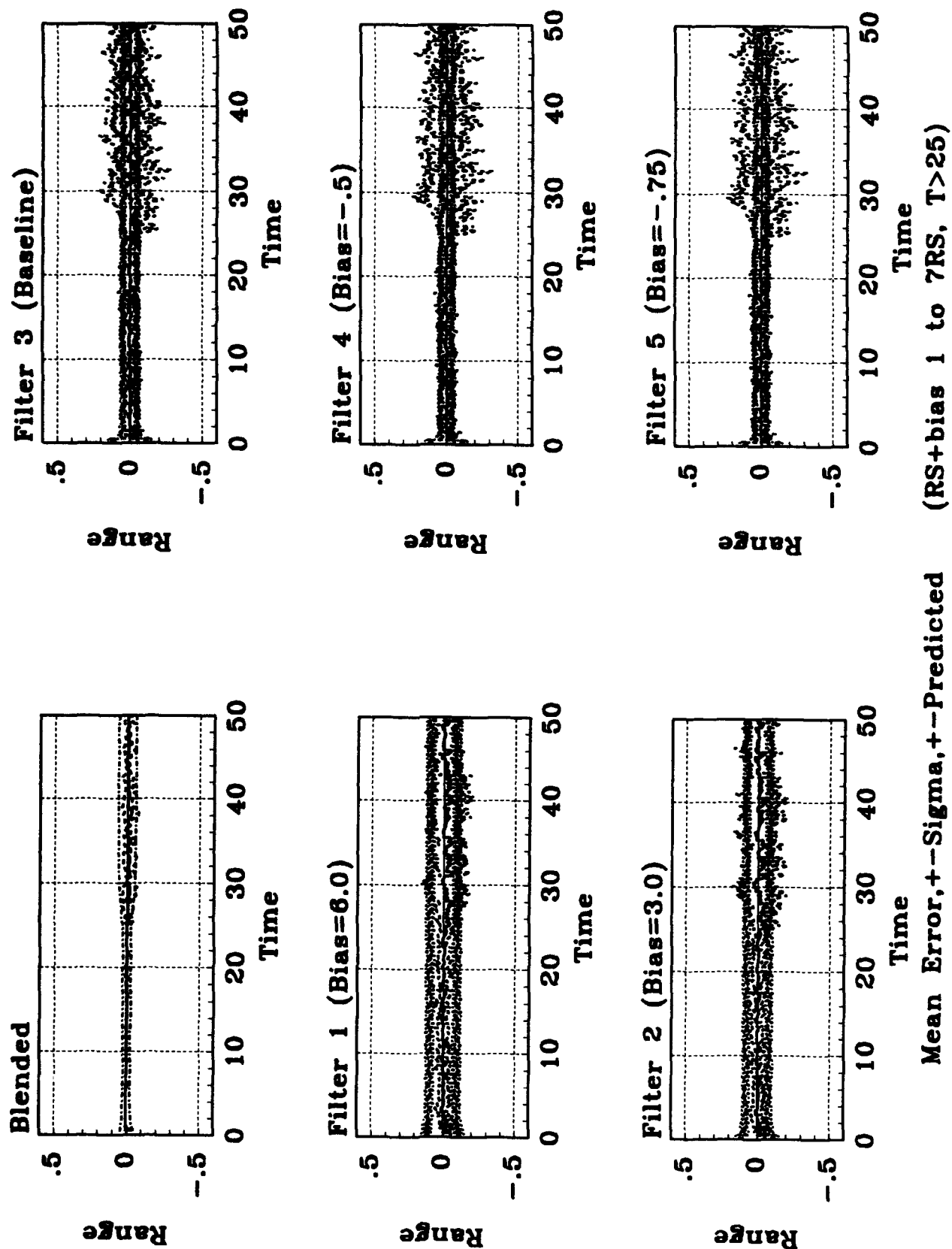


Figure B.4. Case #1 Range

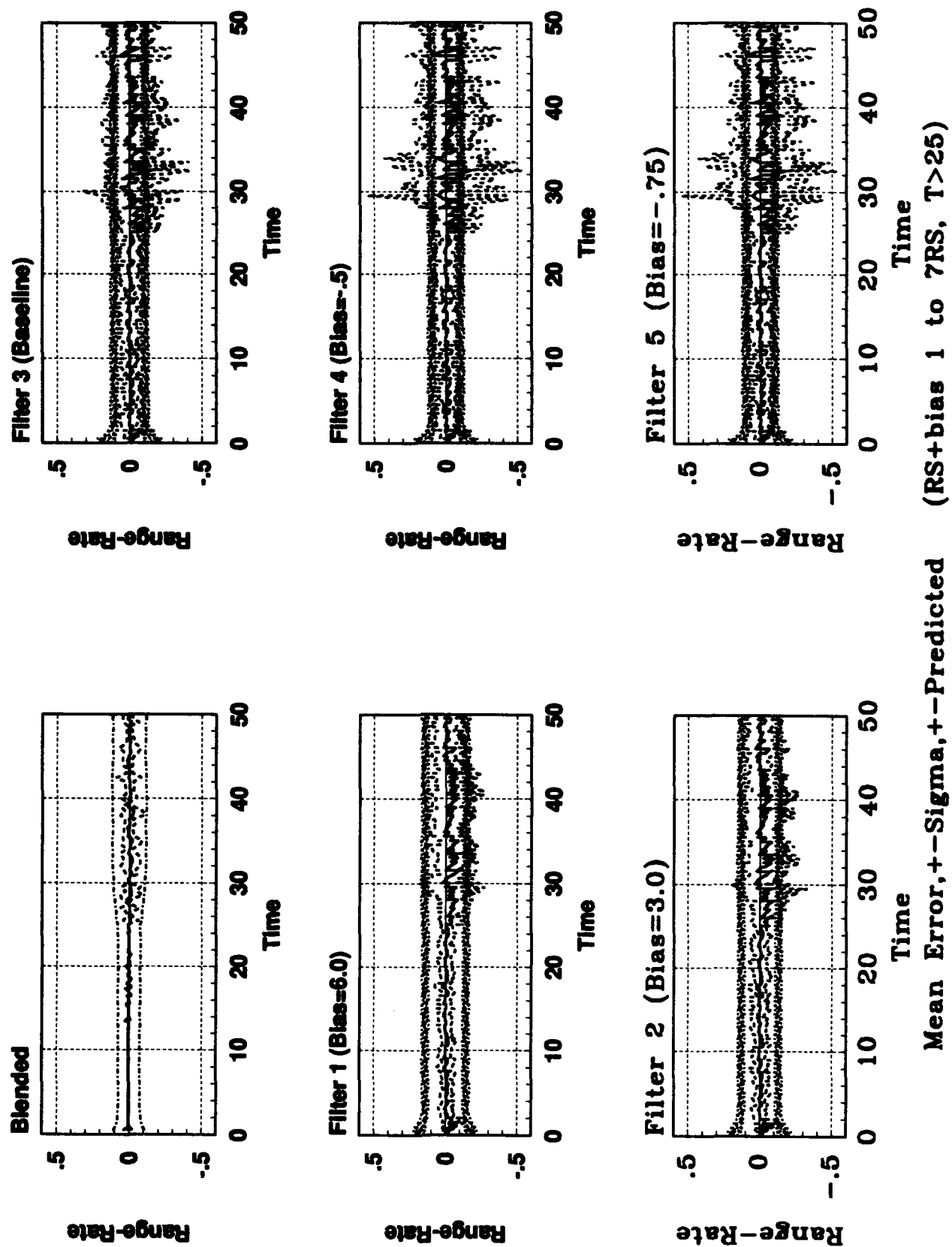


Figure B.5. Case #1 Range-rate

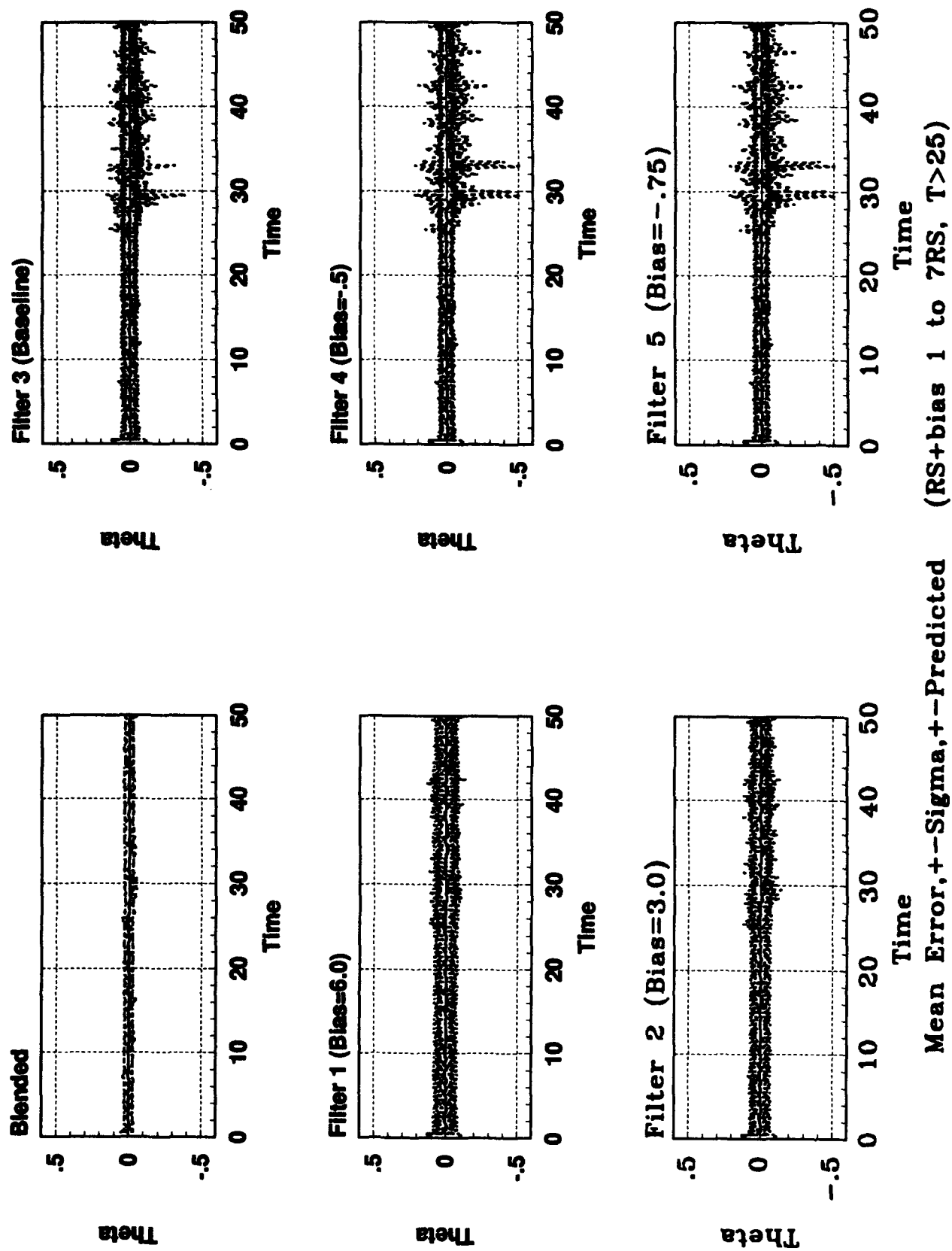
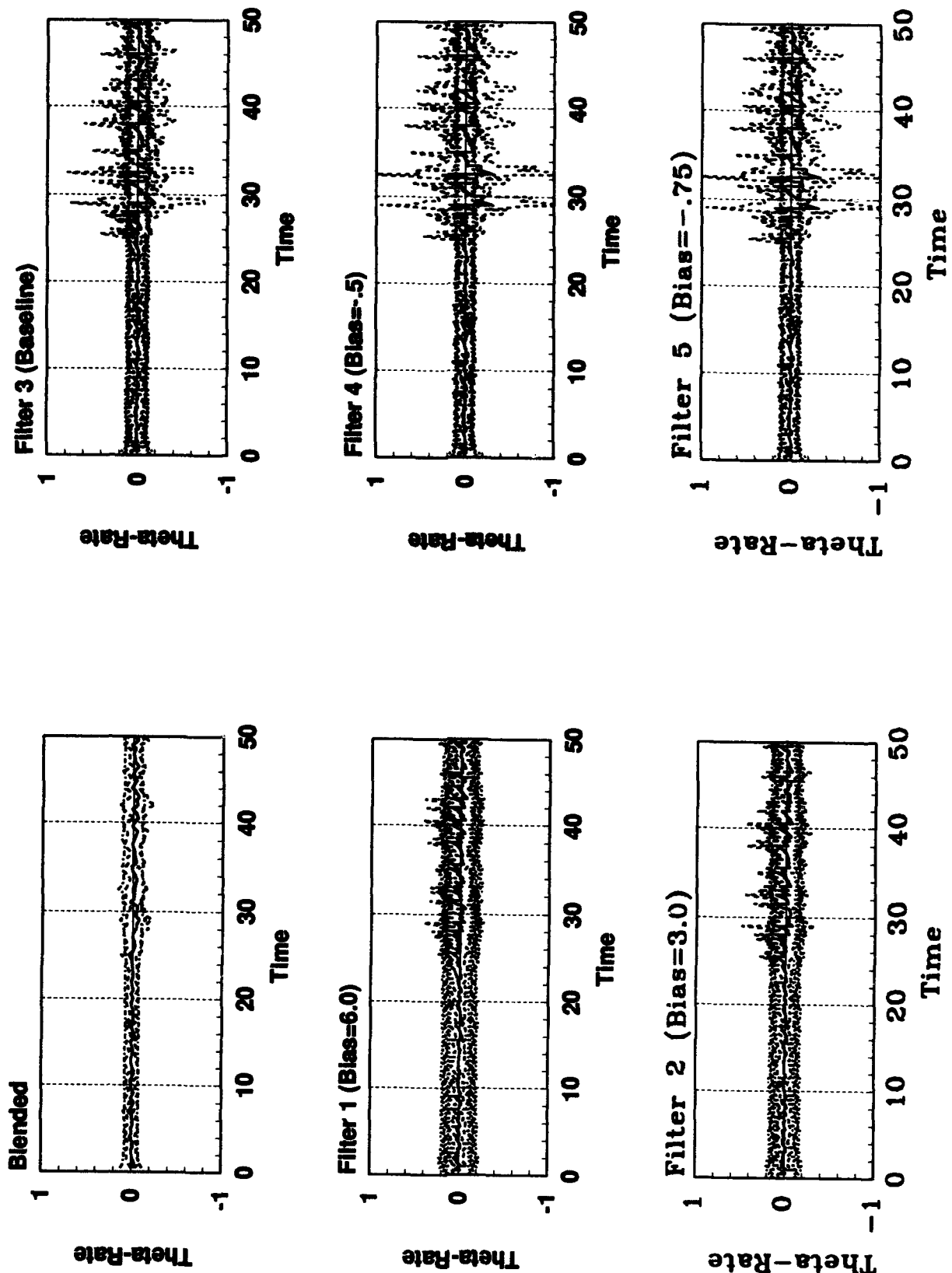


Figure B.6. Case #1 Theta
B-8



Mean Error, +-Sigma, +-Predicted (RS+bias 1 to 7RS, T>25)

Figure B.7. Case #1 Theta-rate
B-9

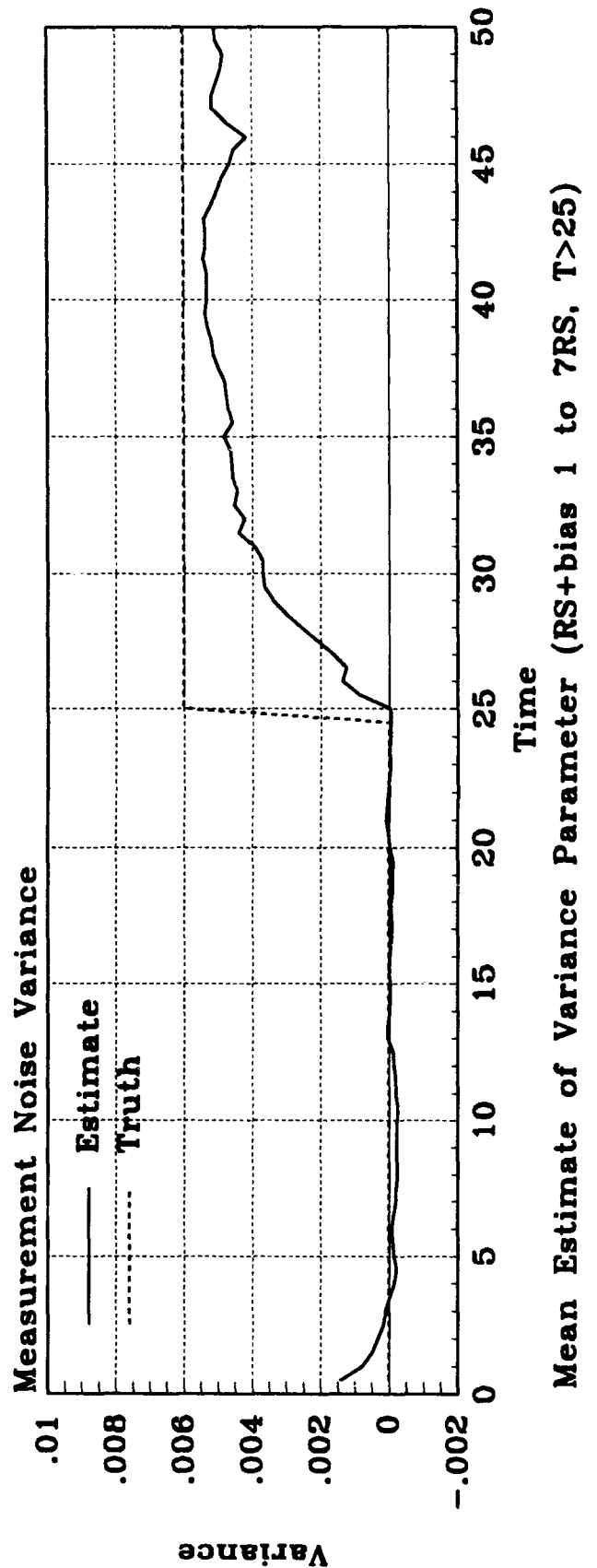
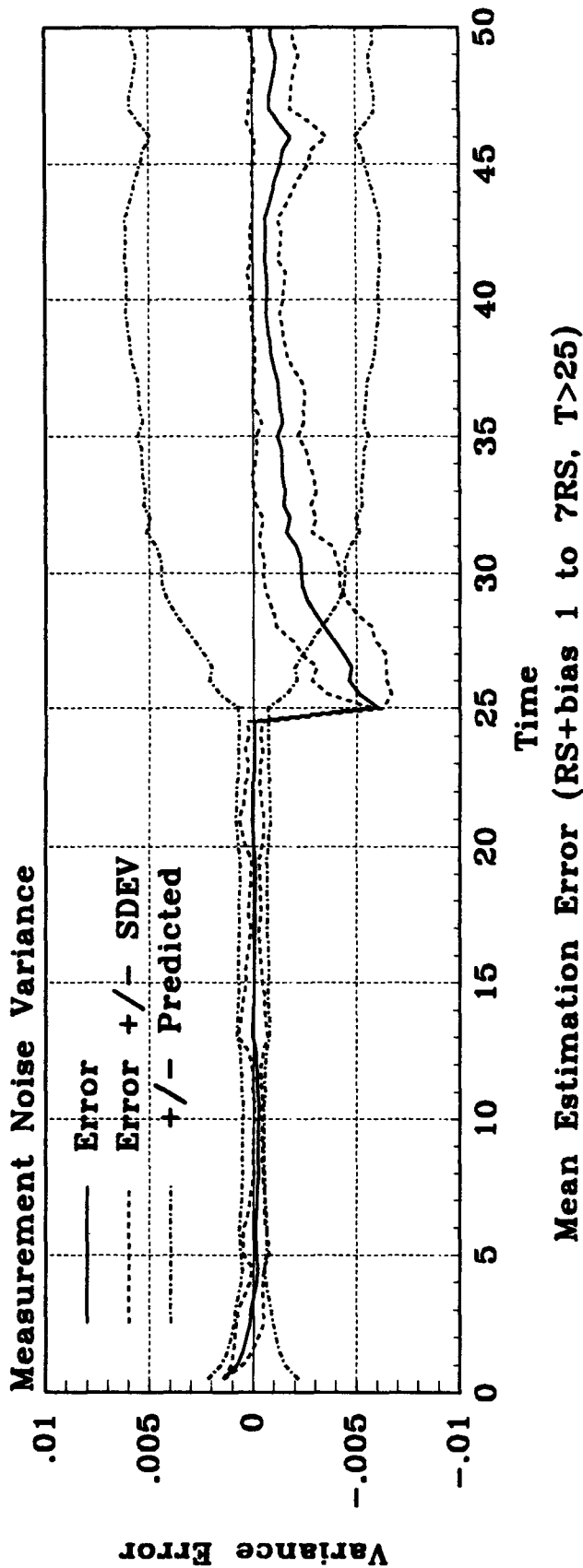


Figure B.8. Case #1 Parameter

B.2 Measurement Noise Ramp during $25 < T < 35$

Table B.2. Orbit Problem Failure Case #2

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
2	Ramp in Measurement Noise	$R_S = R_{S0} + \frac{6R_{S0}(T-25)}{(15)}$ $R_S = R_{S0} + 6R_{S0}$	$25 < T < 35$ $T > 35$	$R_{f0}(1 + BIAS_{R_I})$

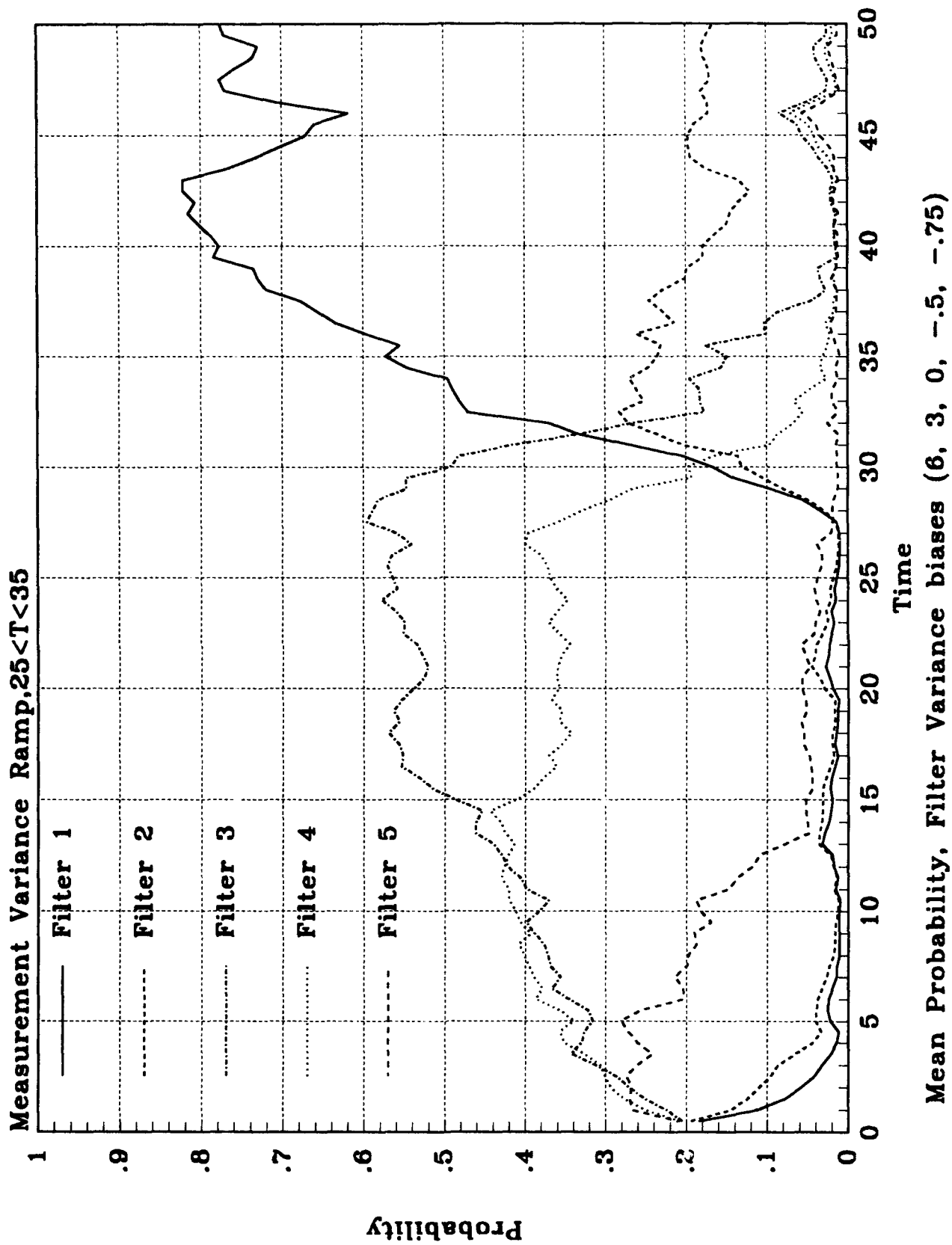


Figure B.9. Case #2 Probabilities

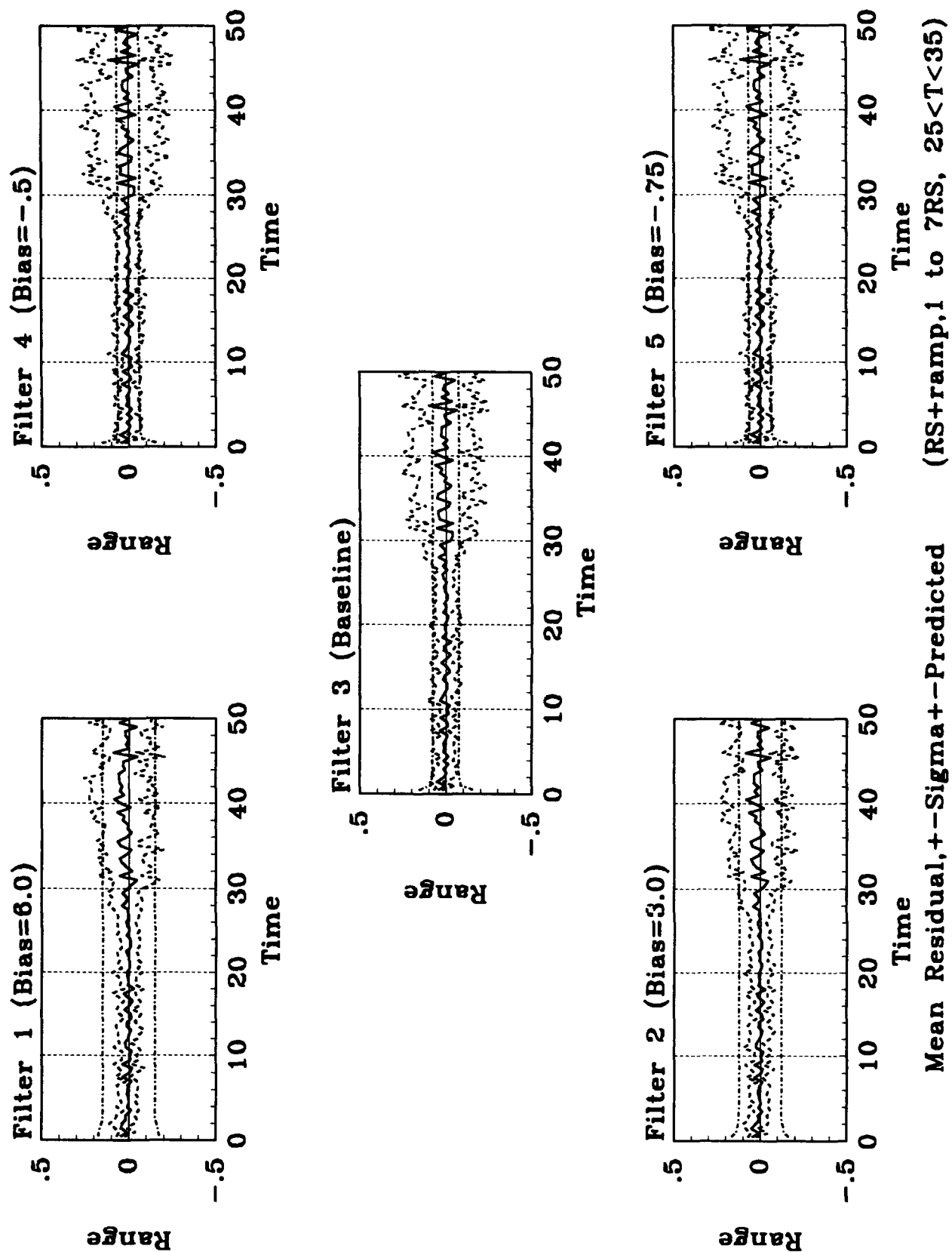


Figure B.10. Case #2 Range Residuals

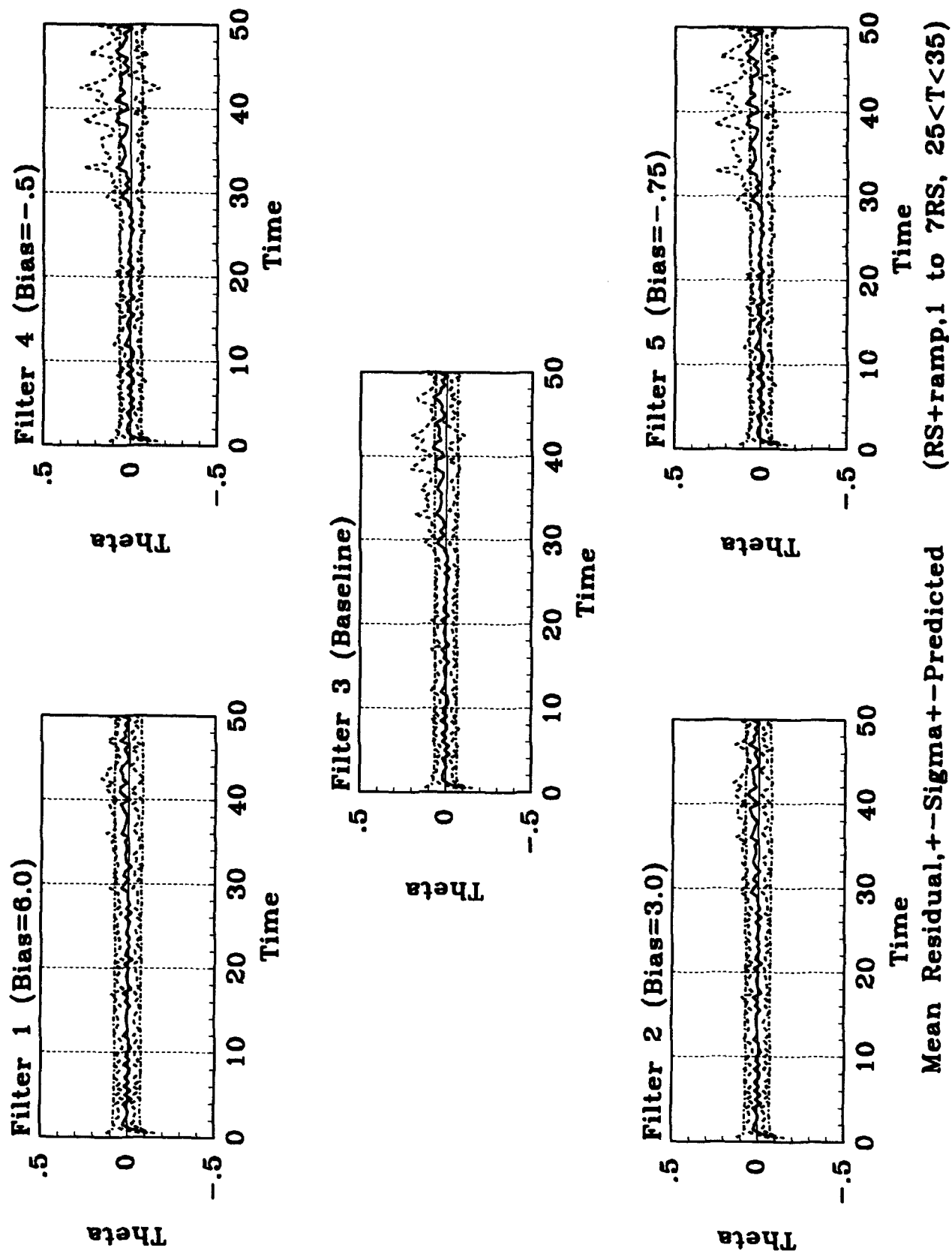


Figure B.11. Case #2 Theta Residuals

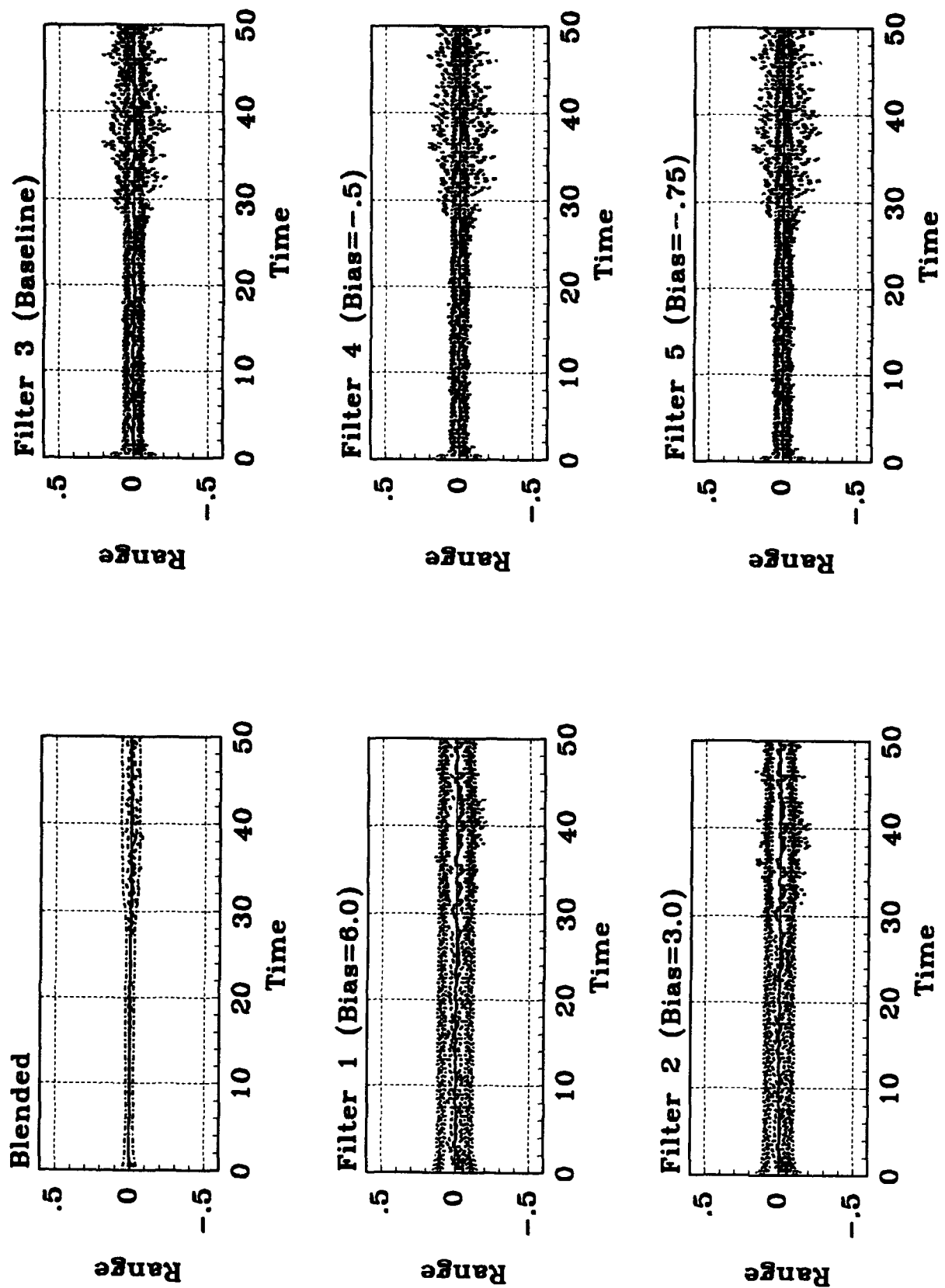


Figure B.12. Case #2 Range

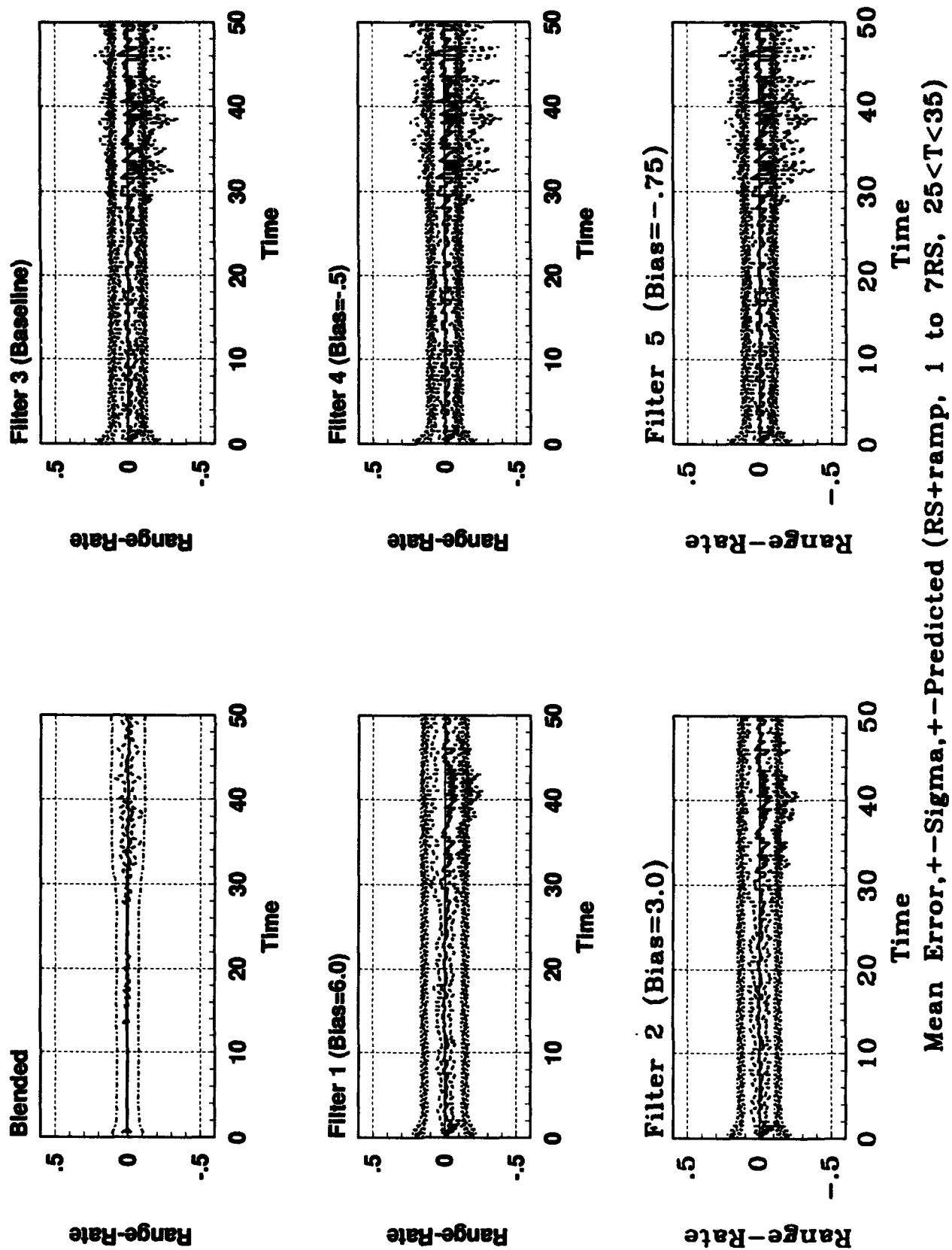


Figure B.13. Case #2 Range-rate
B-16

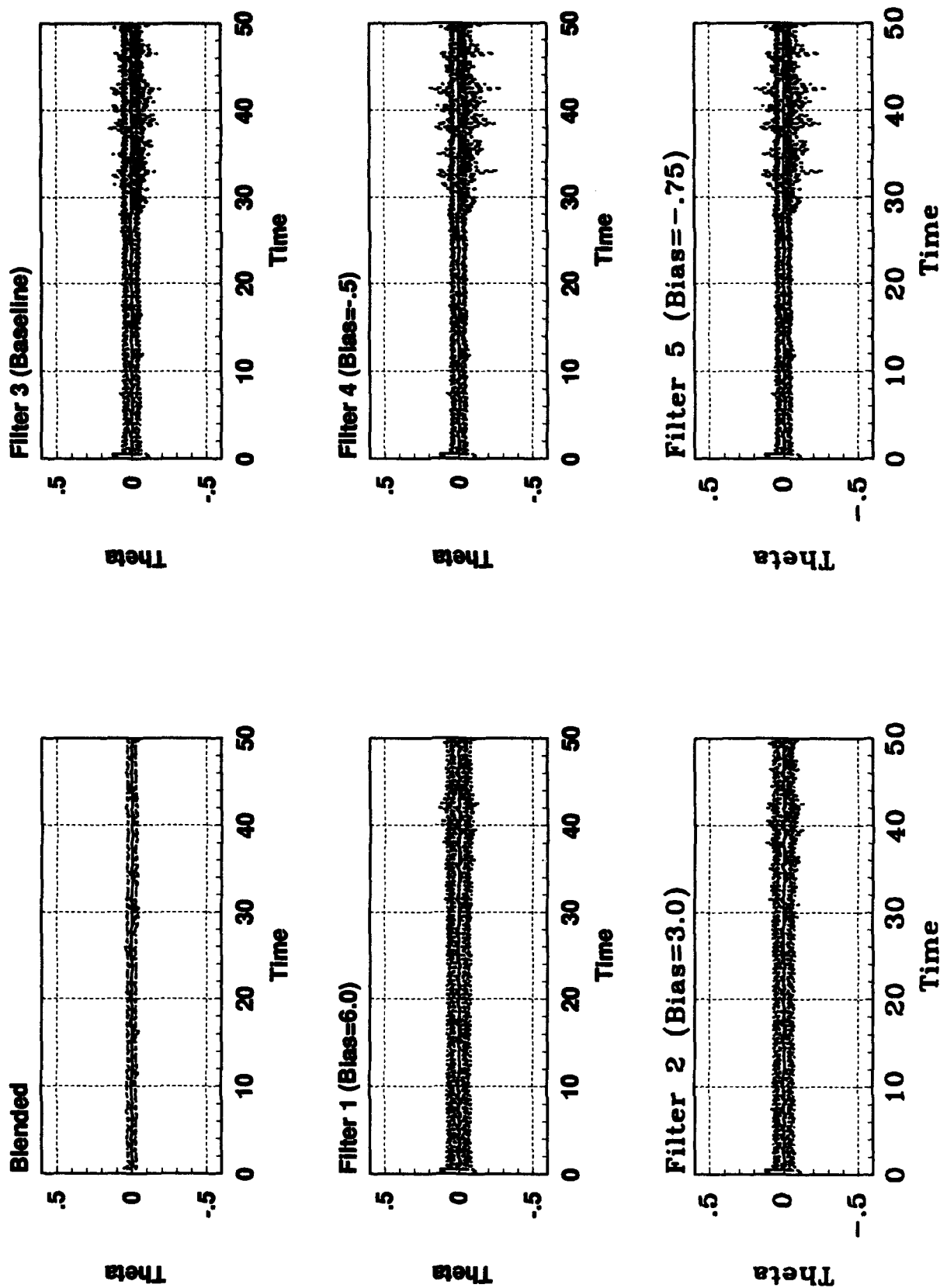


Figure B.14. Case #2 Theta

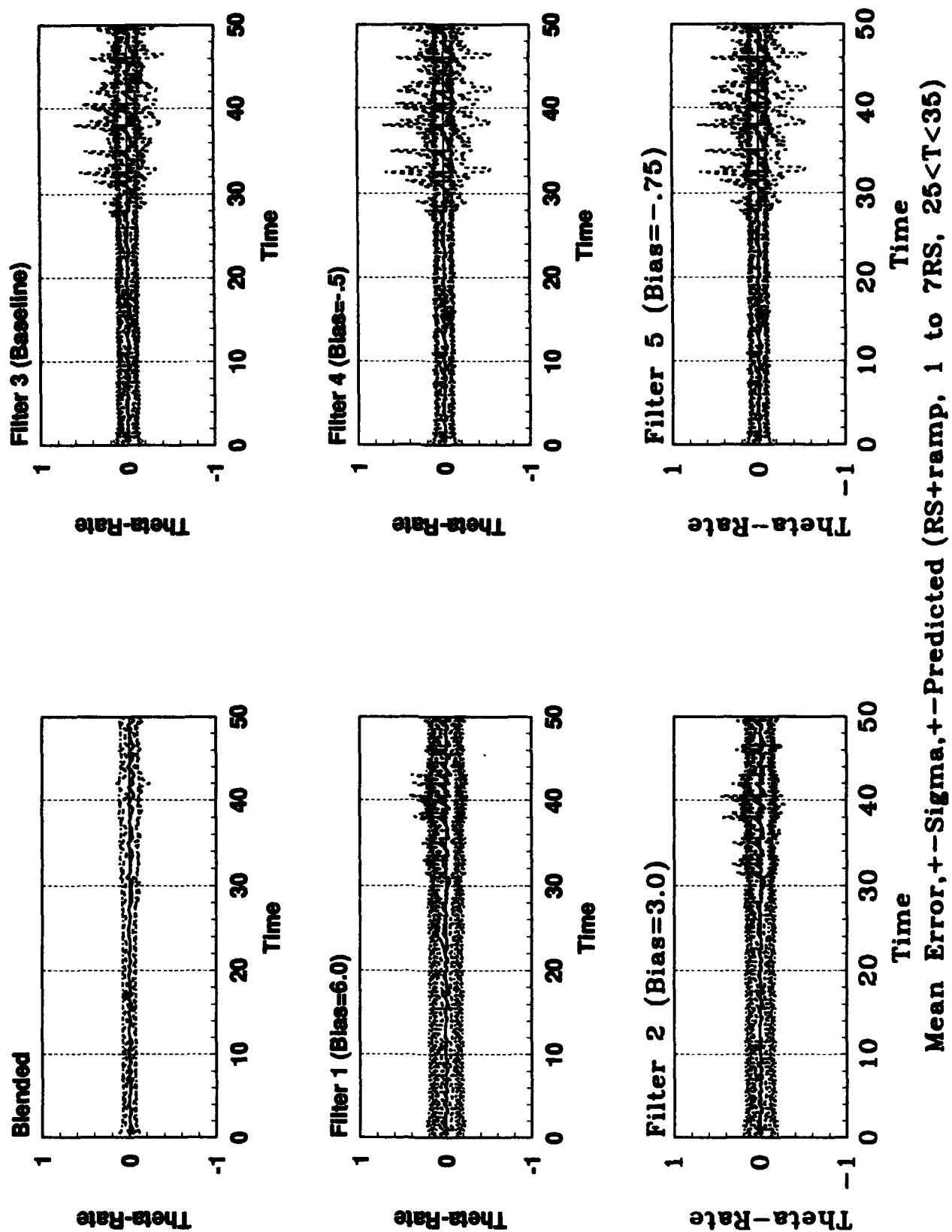
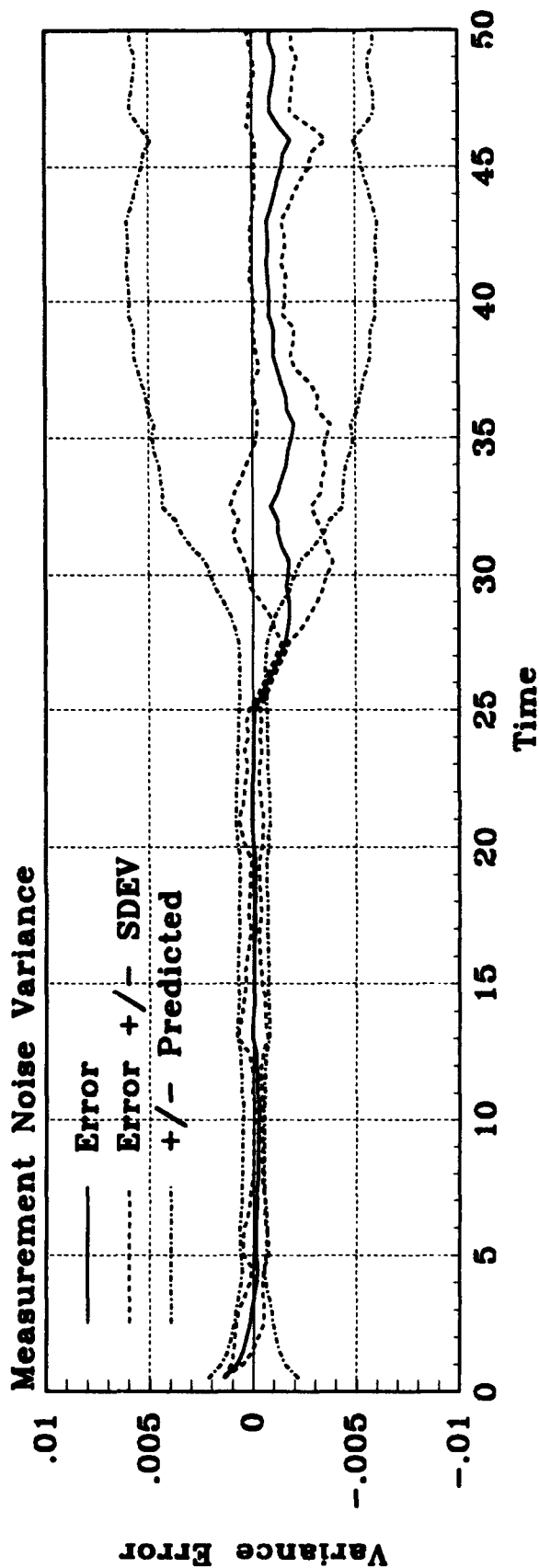
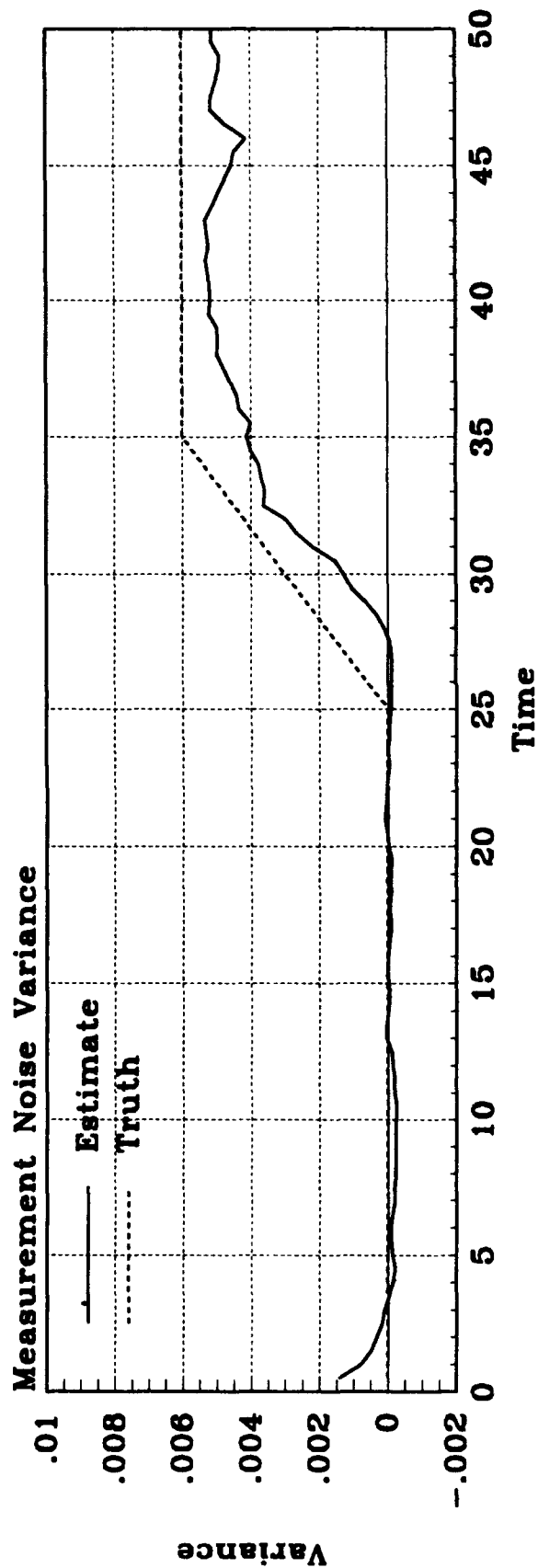


Figure B.15. Case #2 Theta-rate
B-18



Mean Estimation Error (RS+ramp, 1 to 7RS, 25<T<35)



Mean Estimate of Variance Parameter (RS+ramp, 1 to 7RS, 25<T<35)

Figure B.16. Case #2 Parameter

B.3 Measurement Noise Bias during $25 < T < 35$

Table B.3. Orbit Problem Failure Case #3

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
3	Step/Return in Measurement Noise	$R_S = R_{S0}$ $R_S = R_{S0} + 6R_{S0}$	$T < 25, T > 35$ $25 < T < 35$	$R_{f0}(1 + BIAS_{R_f})$

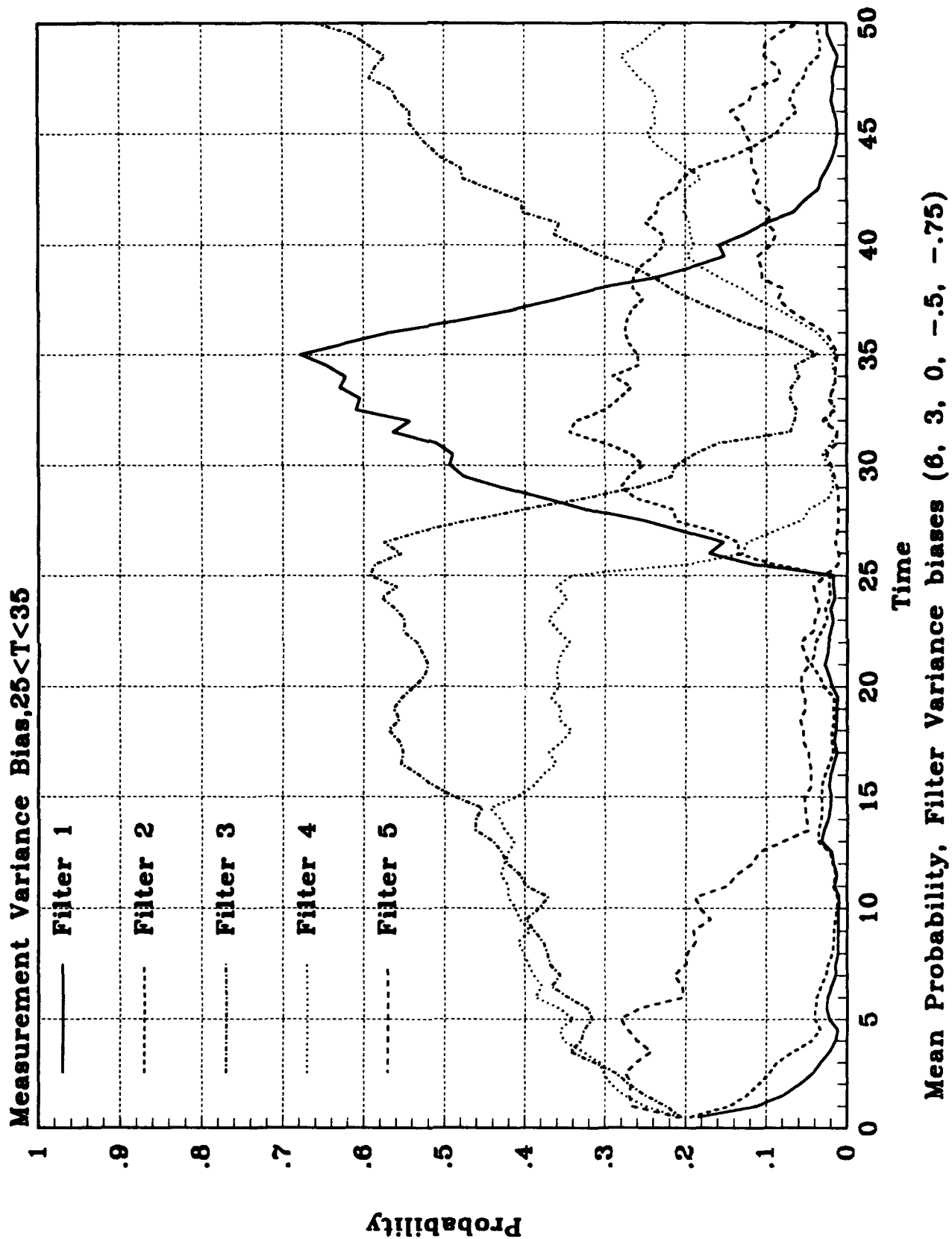


Figure B.17. Case #3 Probabilities

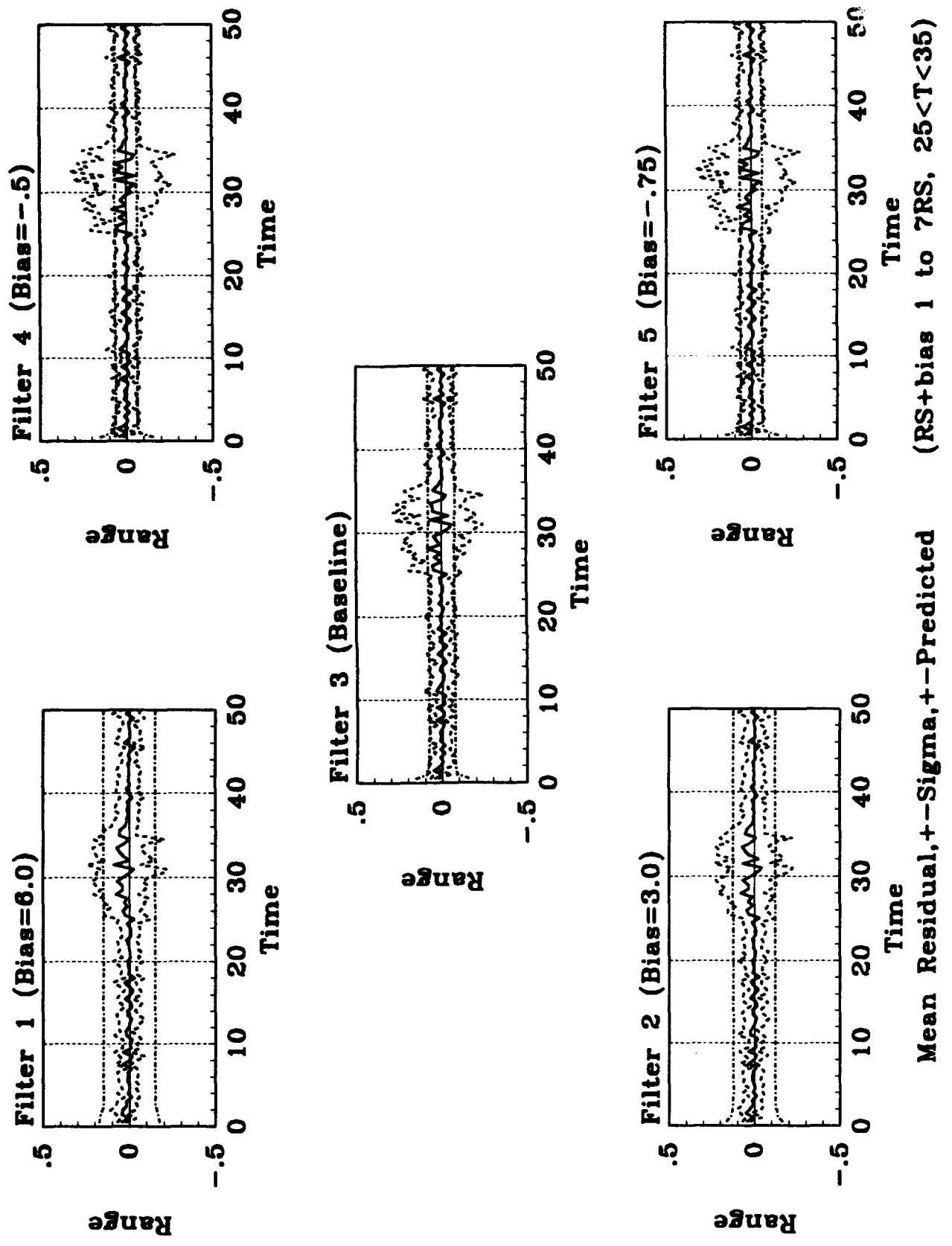


Figure B.18. Case #3 Range Residuals
B-22

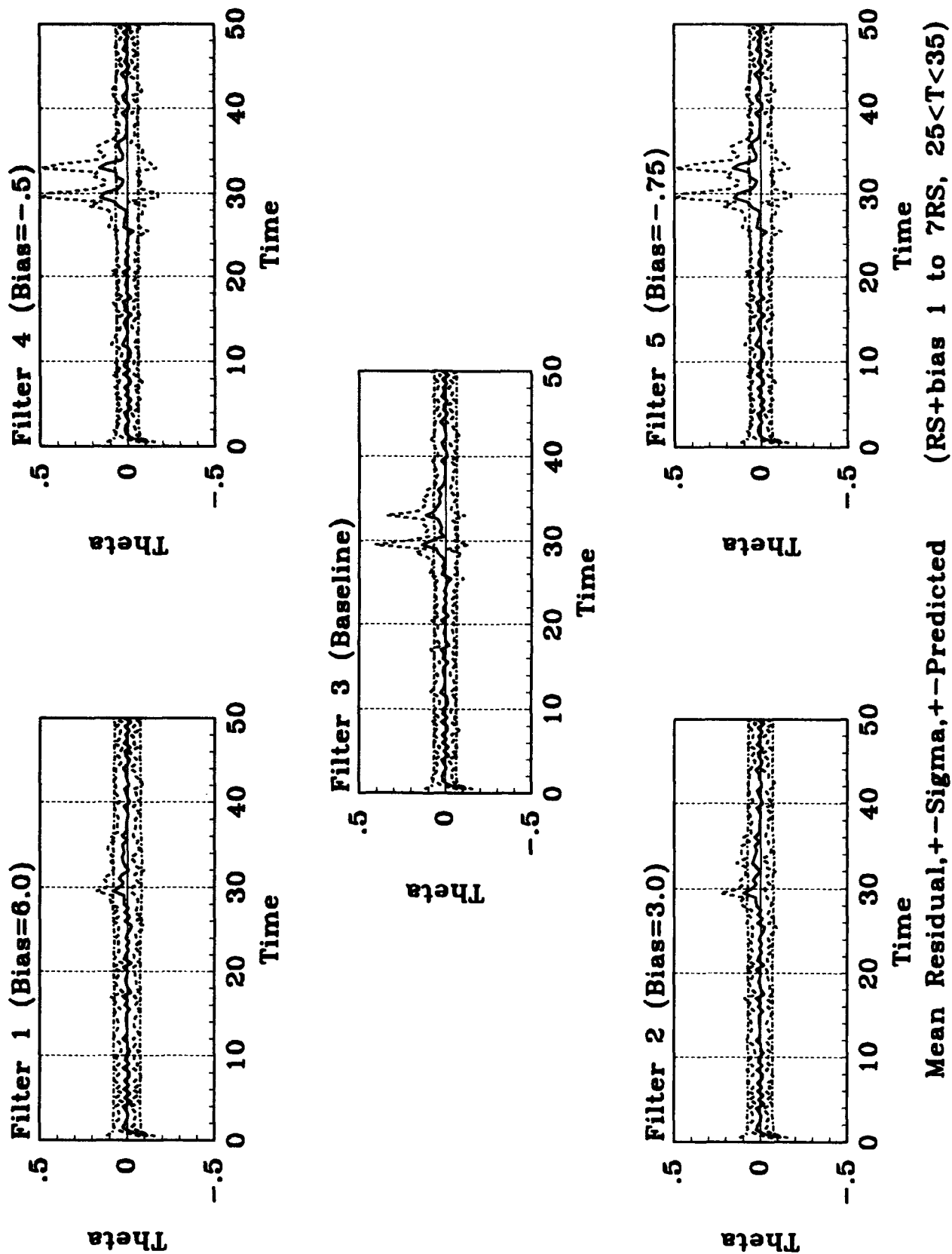
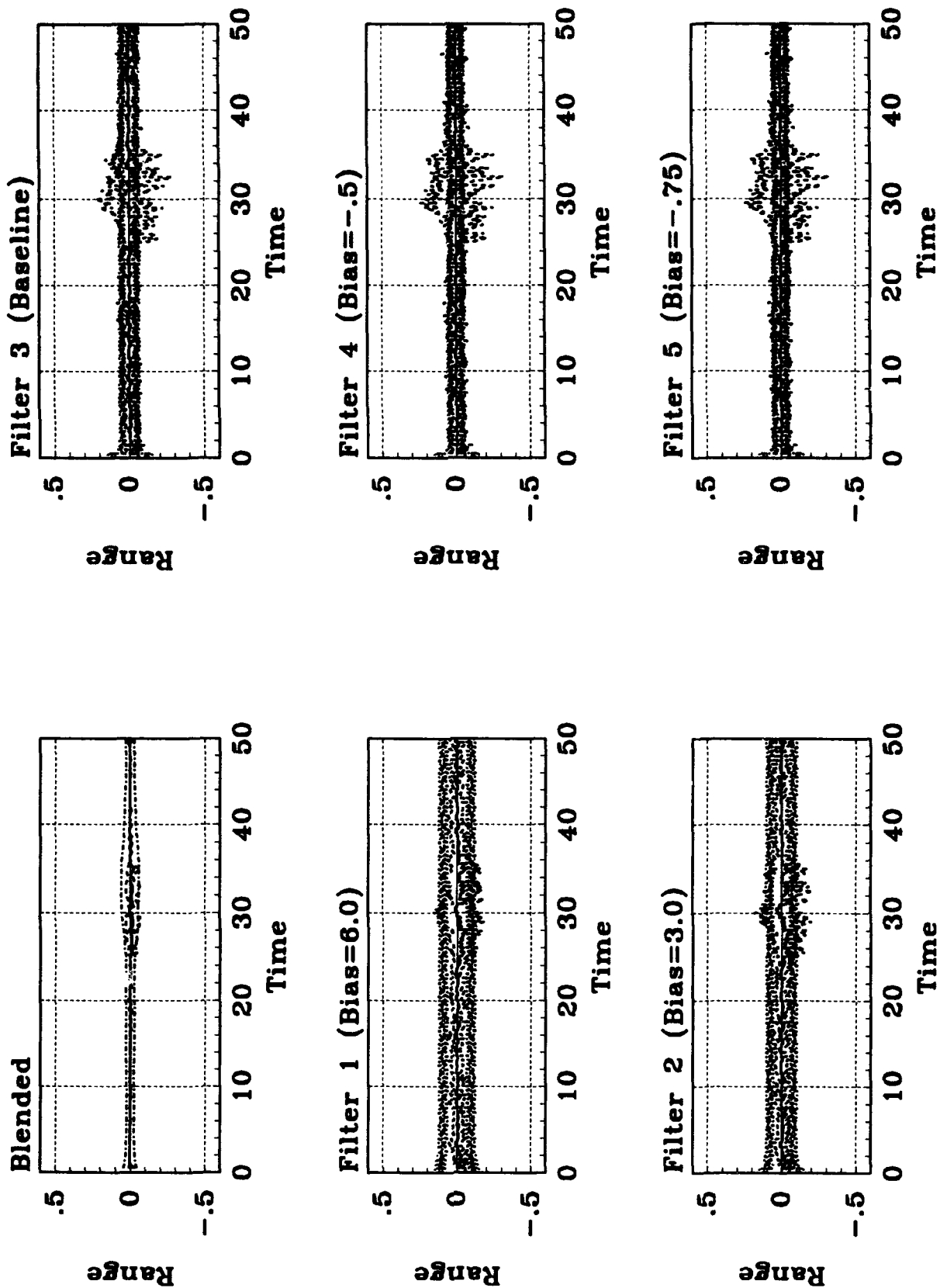
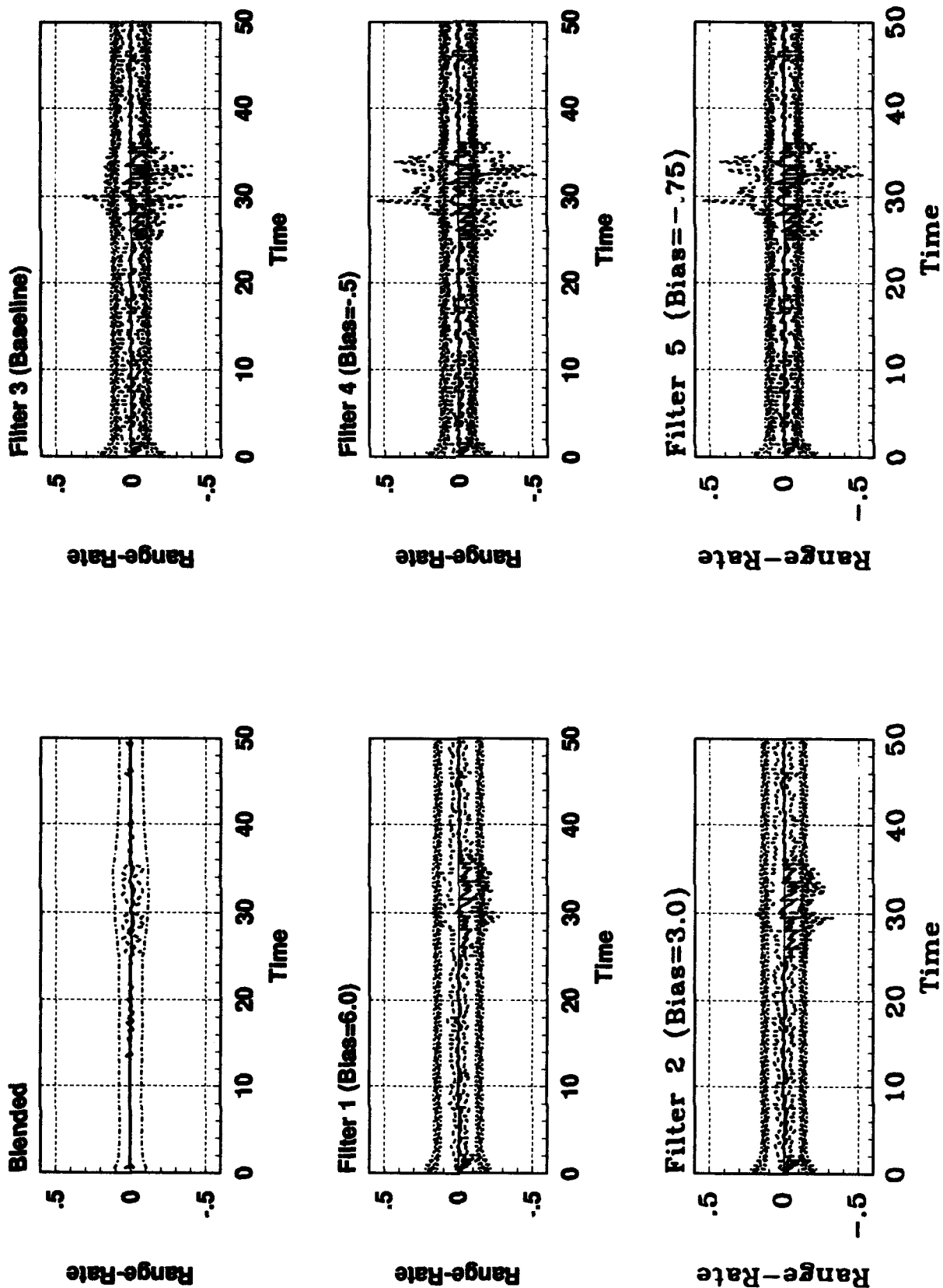


Figure B.19. Case #3 Theta Residuals
B-23



Mean Error, $+ - \text{Sigma}$, $+ - \text{Predicted}$ (RS+bias 1 to 7RS, $25 < T < 35$)

Figure B.20. Case #3 Range



Mean Error, +--Sigma, +--Predicted (RS+bias 1 to 7RS, 25<T<35)

Figure B.21. Case #3 Range-rate

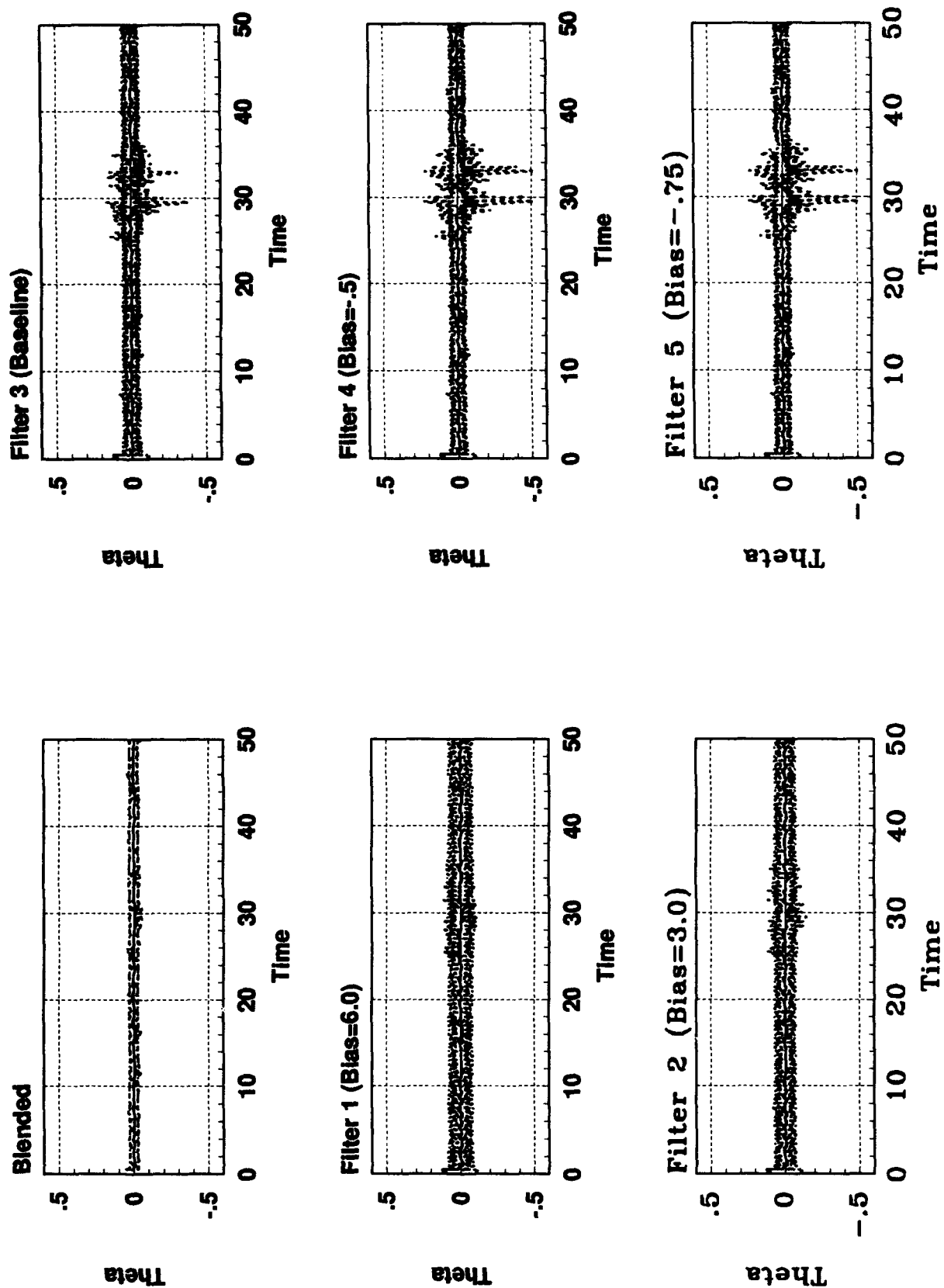


Figure B.22. Case #3 Θ
B-26

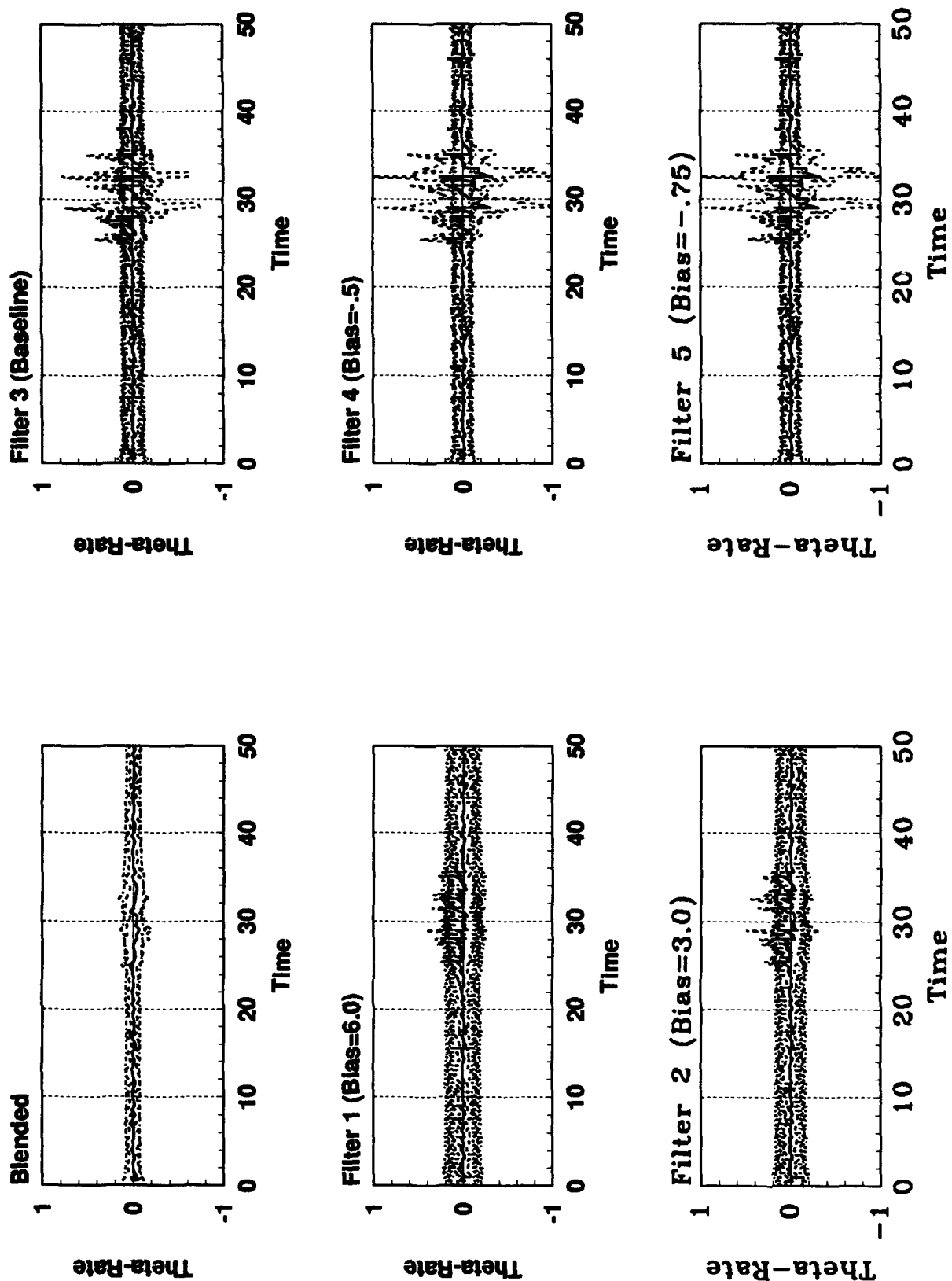
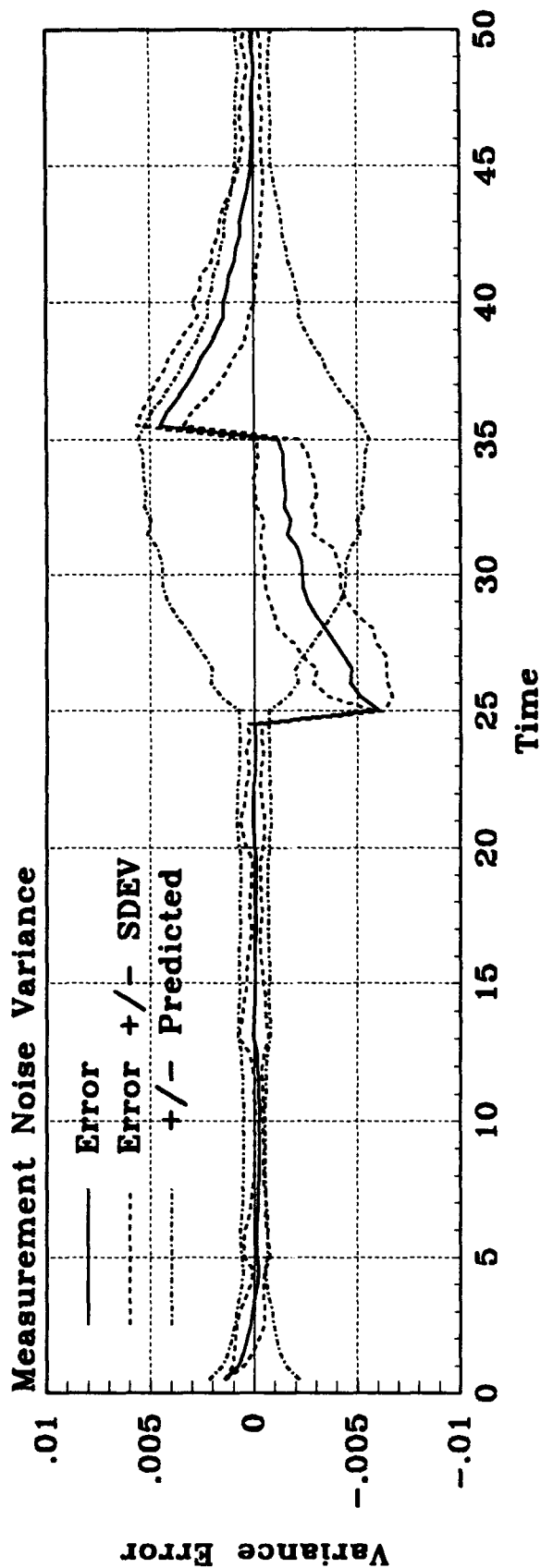
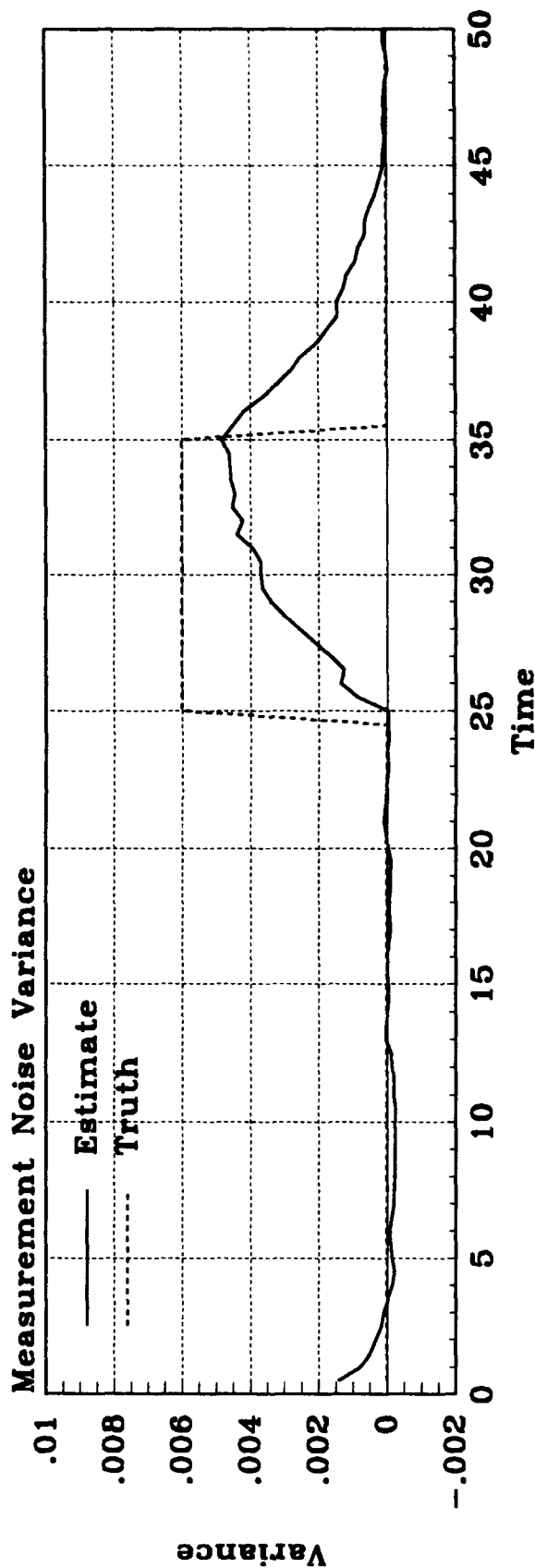


Figure B.23. Case #3 Theta-rate
B-27



Mean Estimation Error (RS+bias 1 to 7RS, 25<T<35)



Mean Estimate of Variance Parameter (RS+bias 1 to 7RS, 25<T<35)

Figure B.24. Case #3 Parameter

B.4 Measurement Bias to .1 for $T > 10$

Table B.4. Orbit Problem Failure Case #4

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
4	Step in Measurement	$Z_S = Z_{S0} + .1Z_{S0}$	$T > 10$	$Z_I + BIAS_{Z_I}$

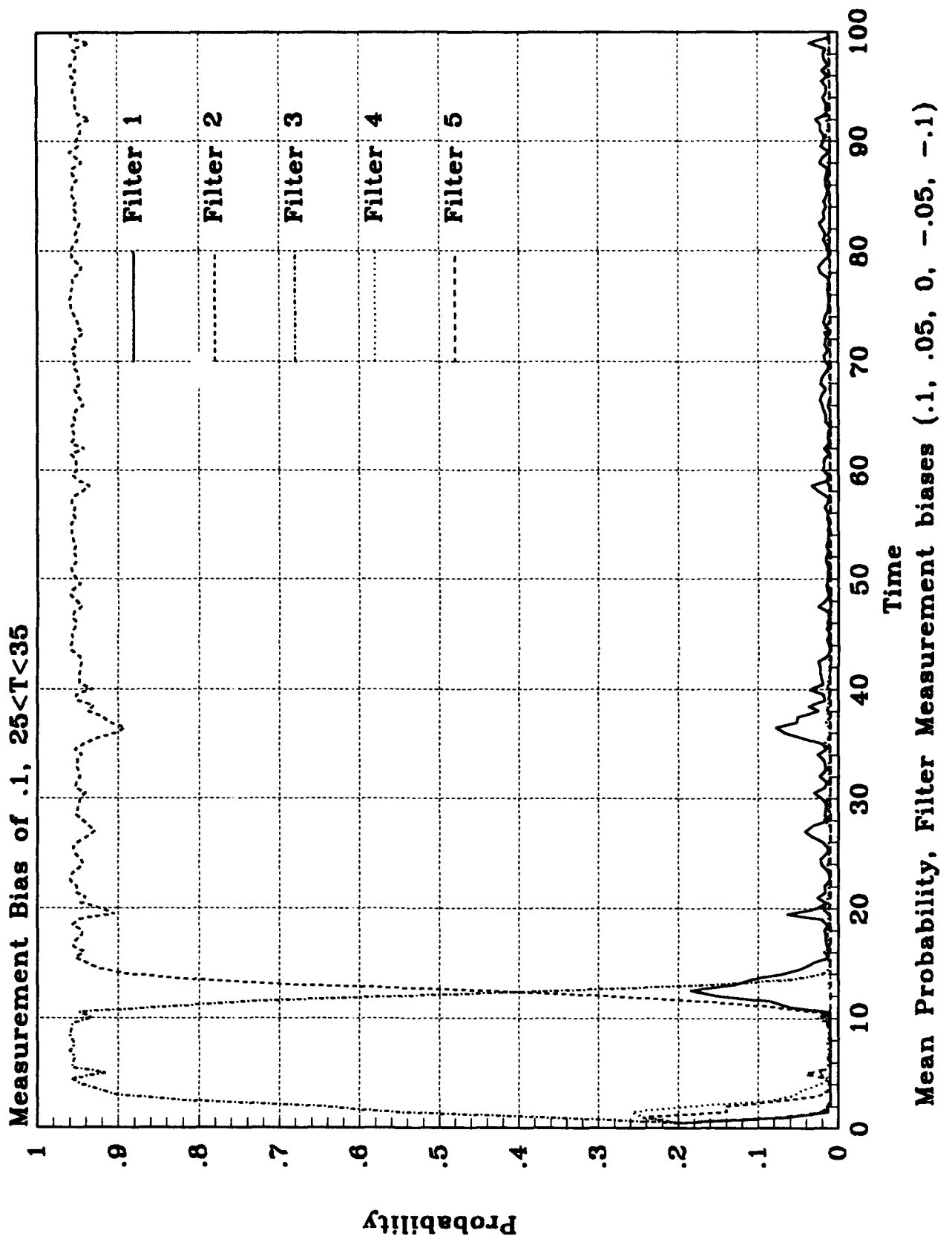


Figure B.25. Case #4 Probabilities
B-30

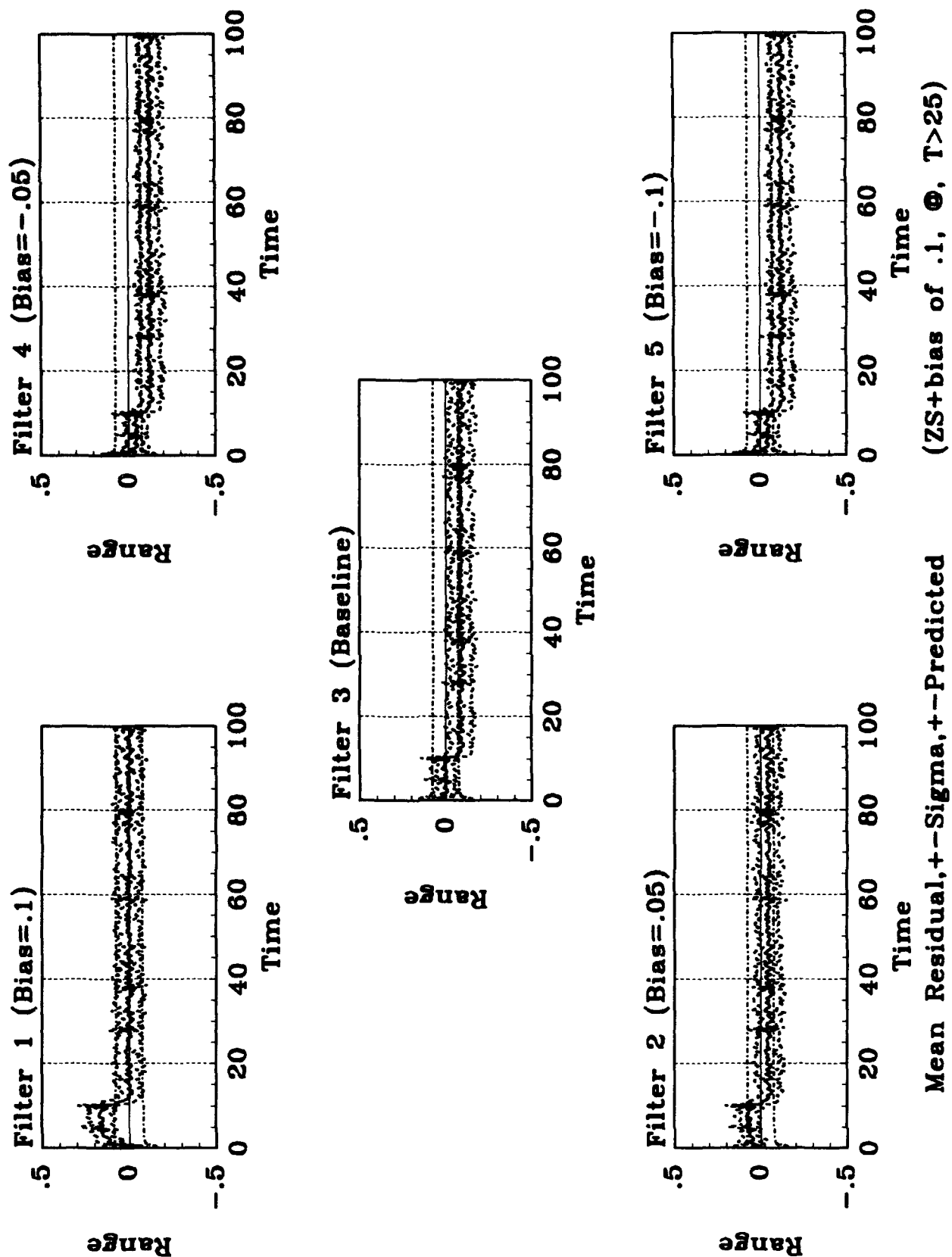


Figure B.26. Case #4 Range Residuals

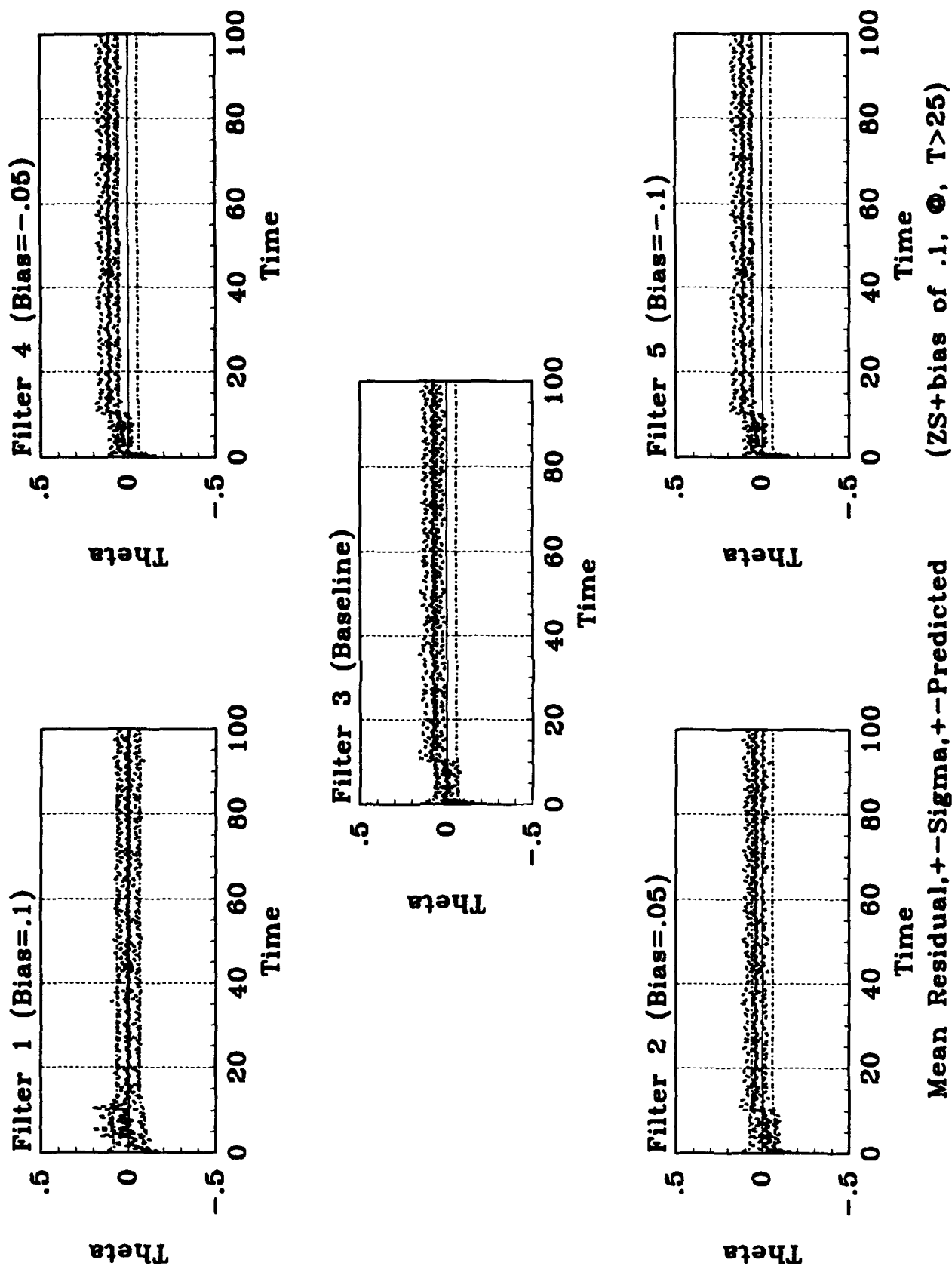


Figure B.27. Case #4 Theta Residuals

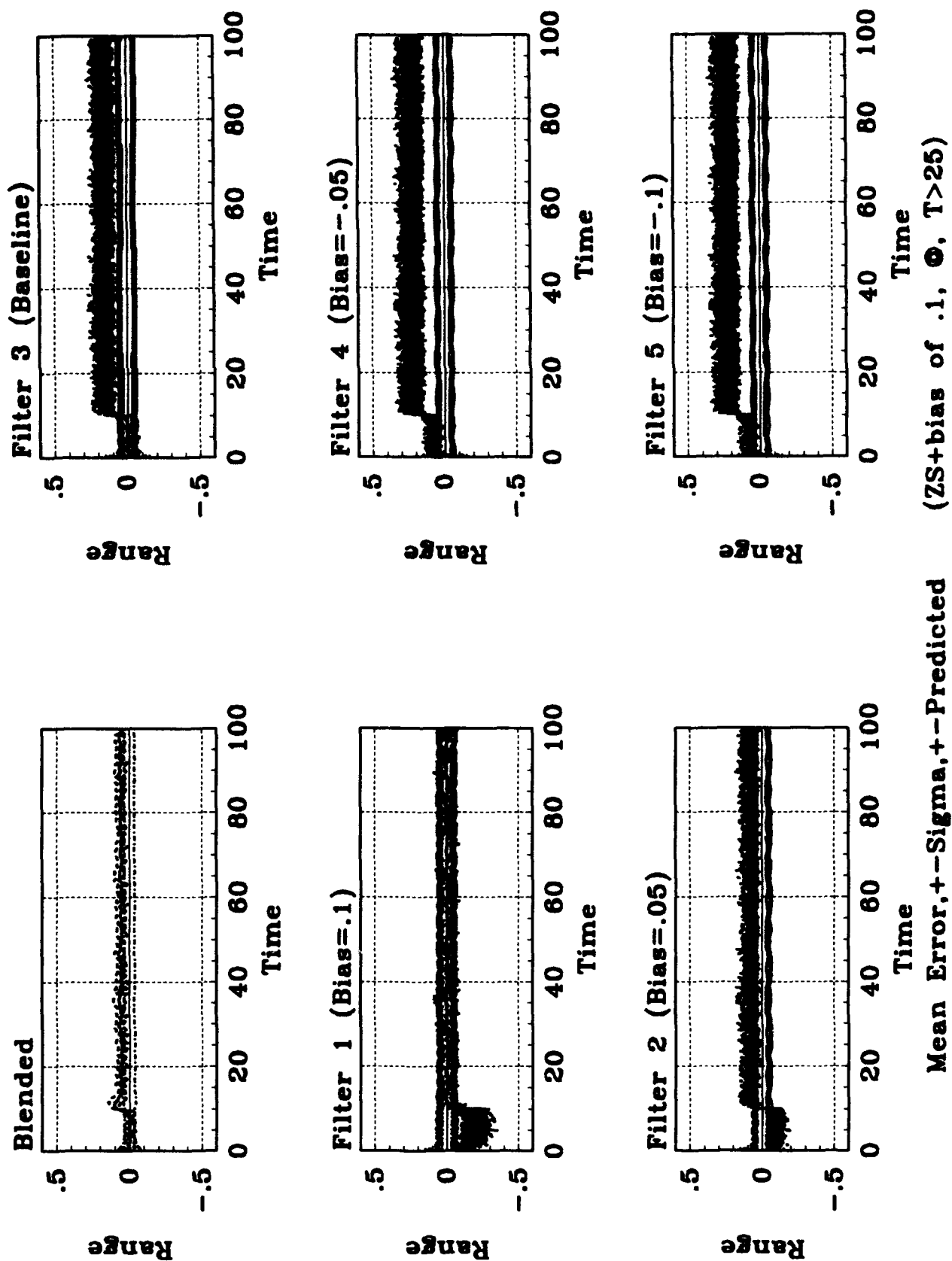


Figure B.28. Case #4 Range
B-33

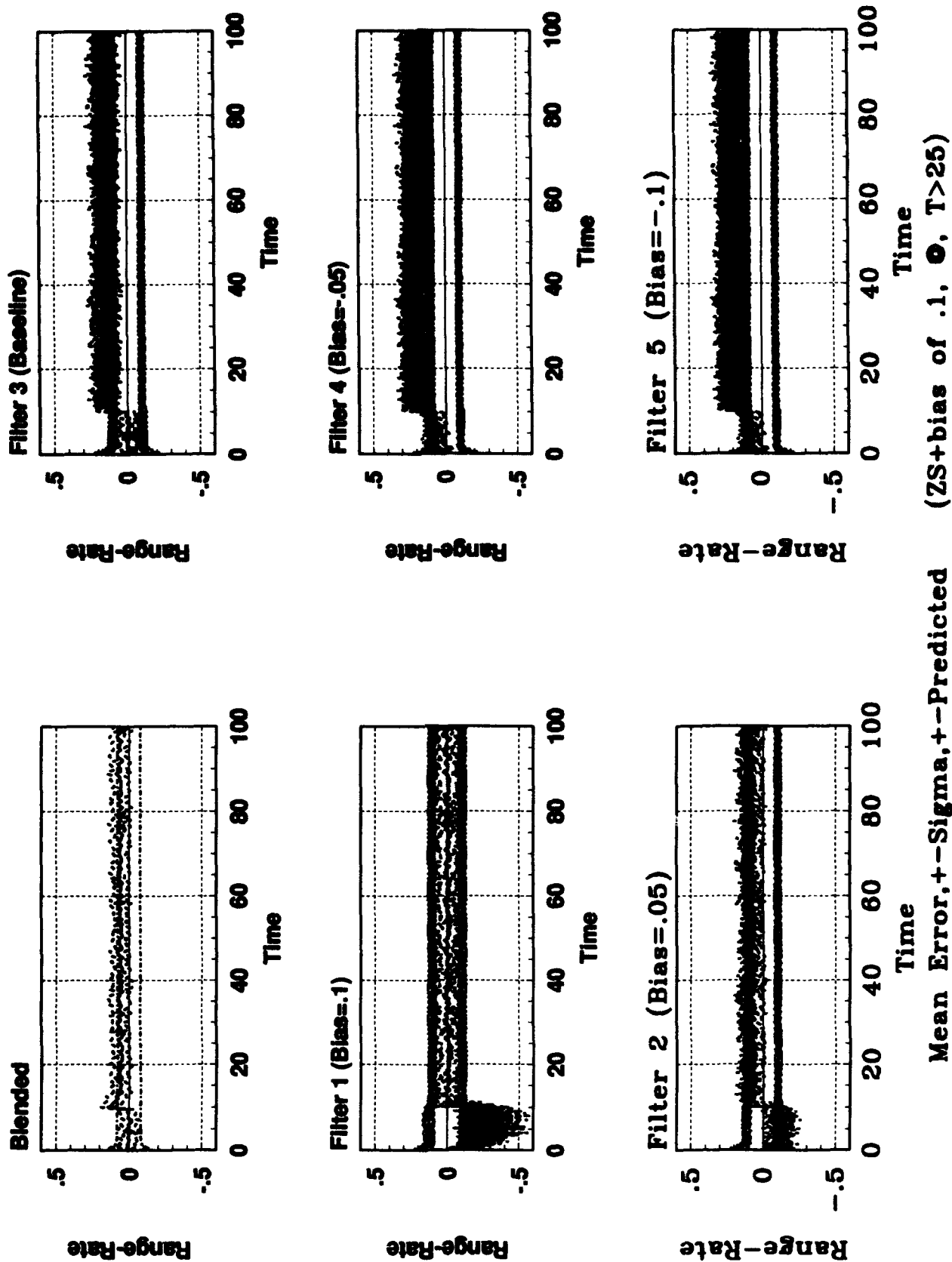
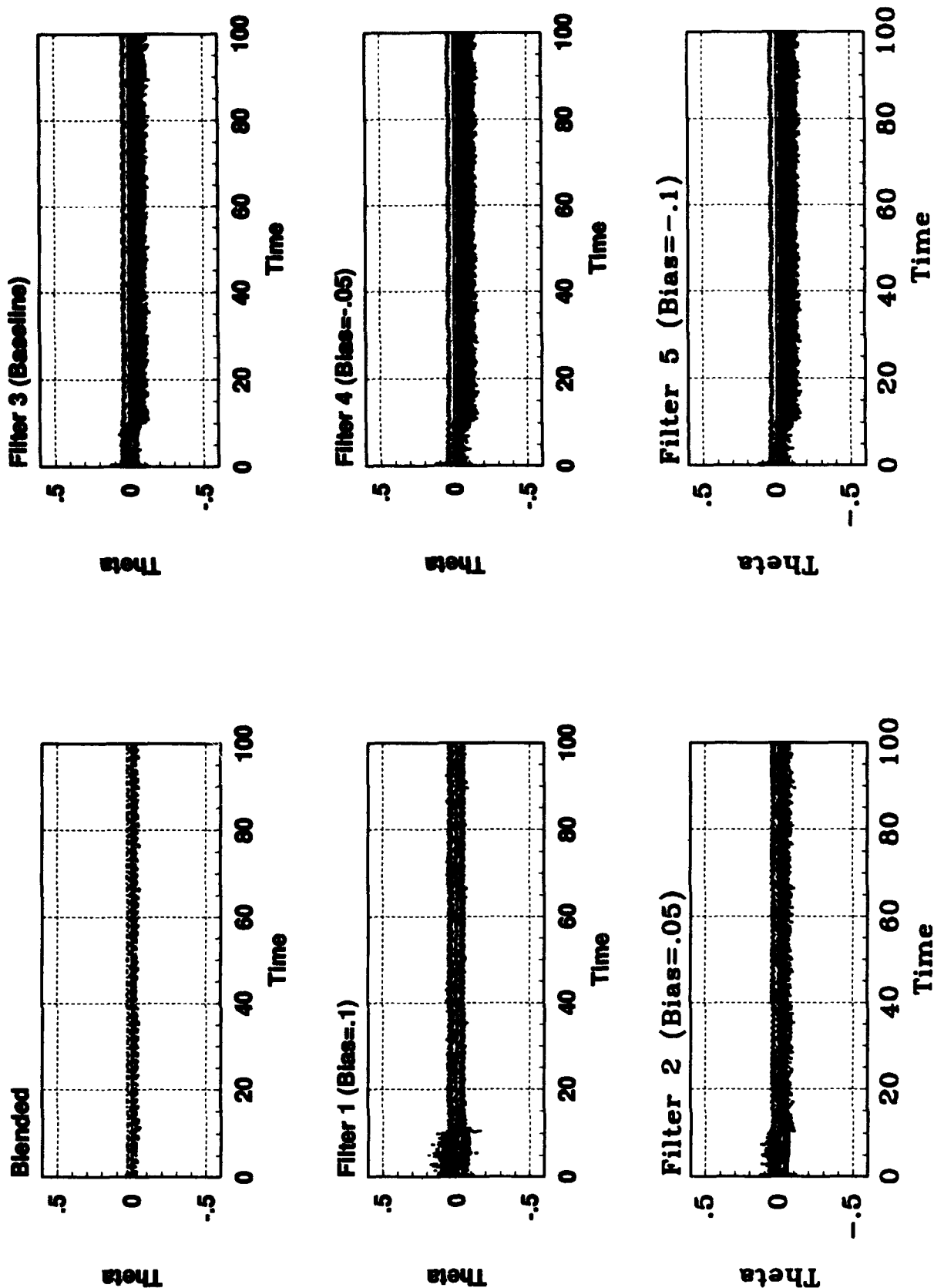


Figure B.29. Case #4 Range-rate



Mean Error, +-Sigma, +-Predicted (ZS+bias of .1, Θ , T>25)

Figure B.30. Case #4 Theta
B-35

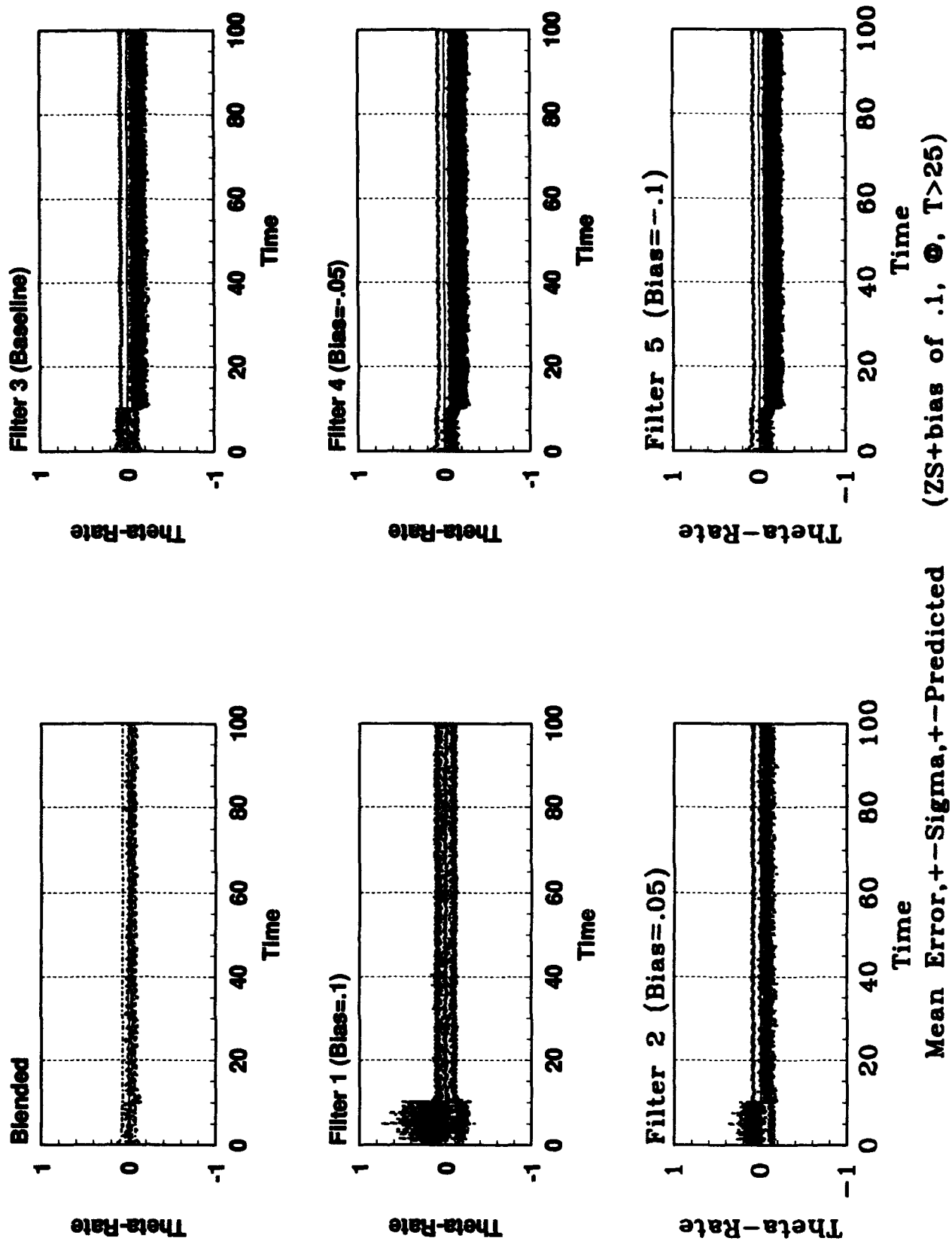
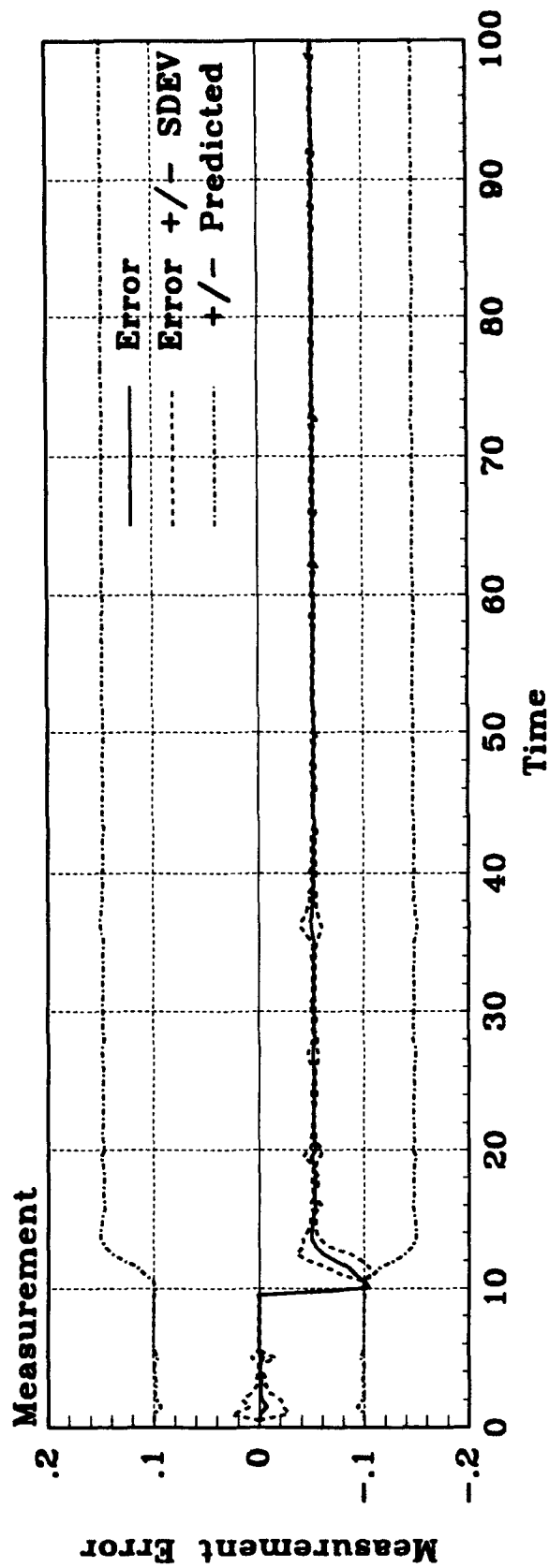
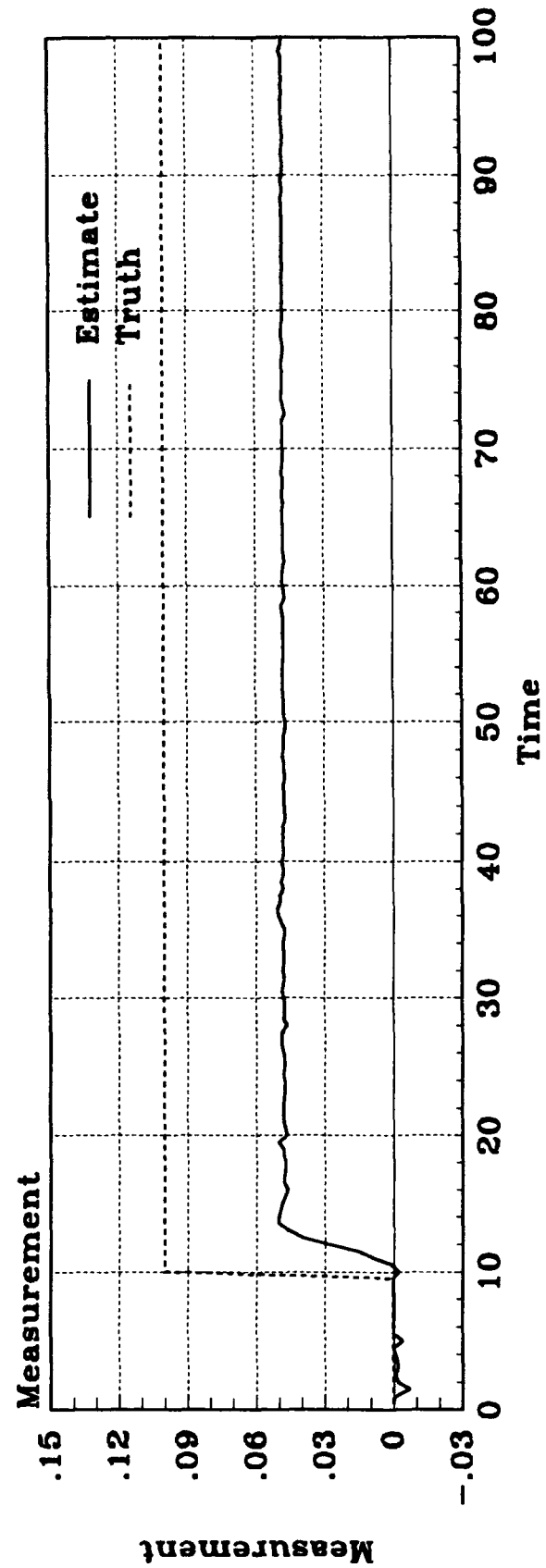


Figure B.31. Case #4 Theta-rate
B-36



Mean Estimation Error (ZS+bias of .1, Θ , T>25)



Mean Estimate of Measurement Parameter(ZS+bias of .1, Θ , T>25)

Figure B.32. Case #4 Parameter
B-37

B.5 Measurement Ramp up to .1 for $10 < T < 20$

Table B.5. Orbit Problem Failure Case #5

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
5	Ramp in Measurement	$Z_S = Z_{S0} + \frac{.1Z_{S0}(T-10)}{(10)}$ $Z_S = Z_{S0} + .1Z_{S0}$	$10 < T < 20$ $T > 20$	$Z_f + BIAS_{Z_f}$

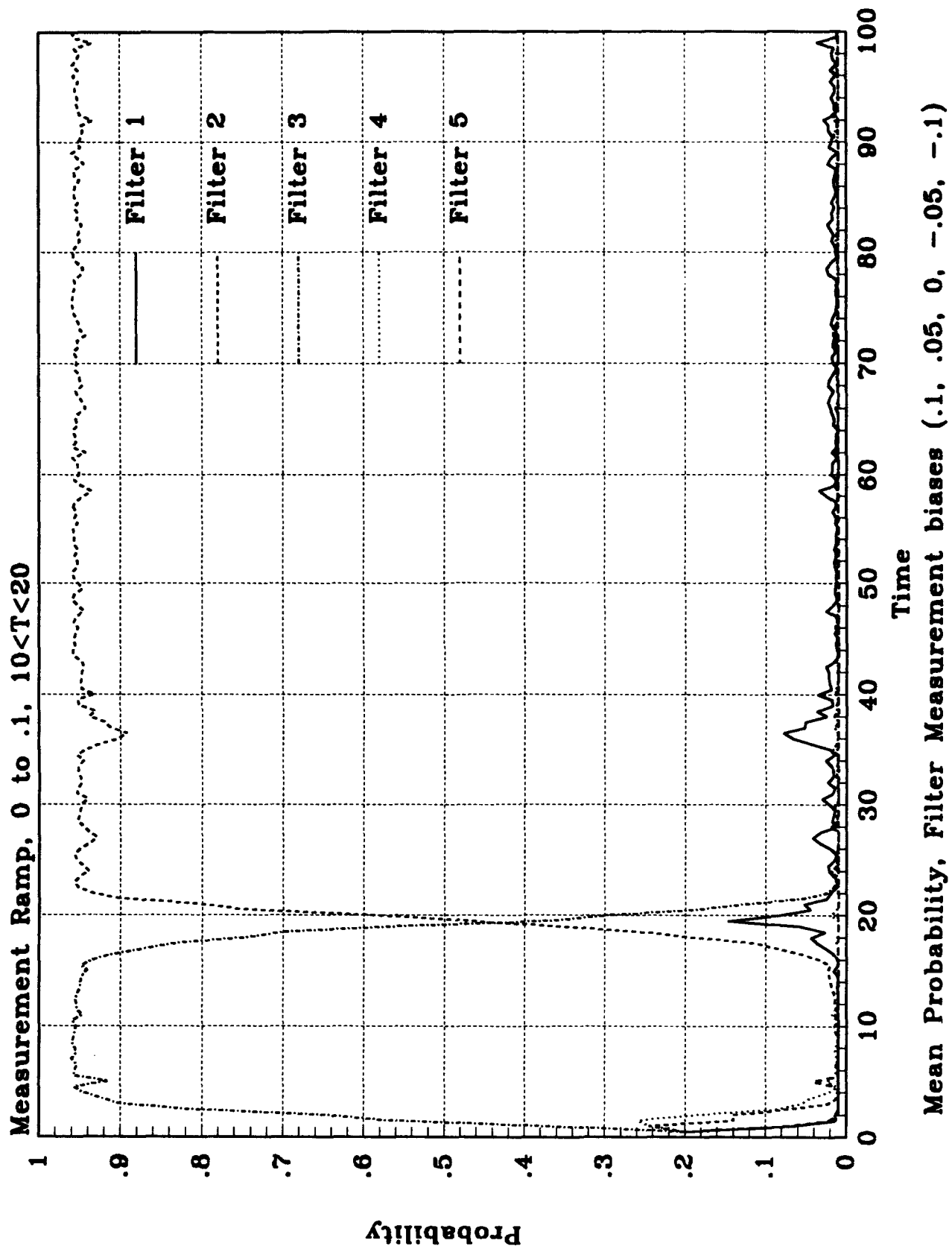


Figure B.33. Case #5 Probabilities

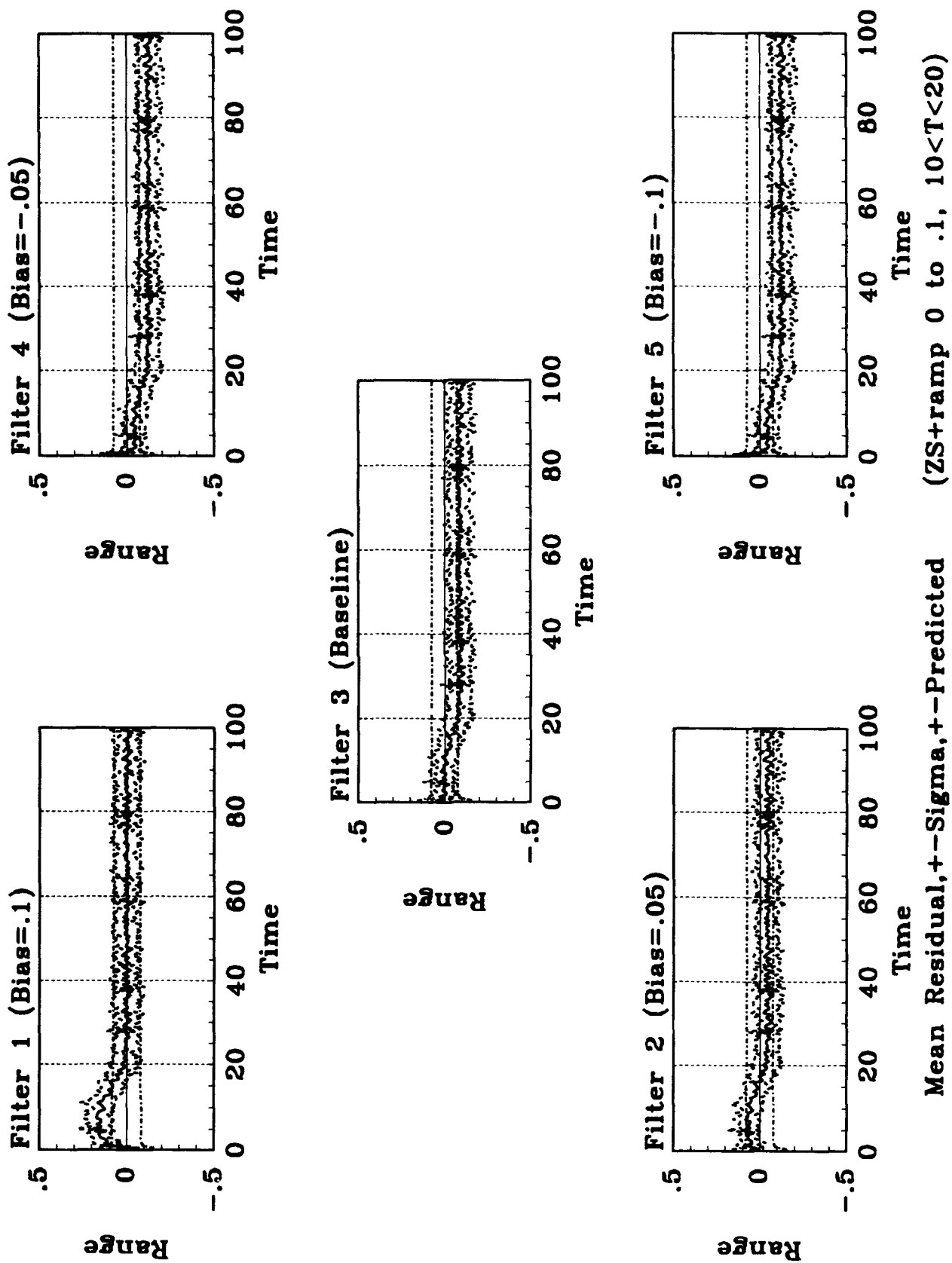


Figure B.34. Case #5 Range Residuals

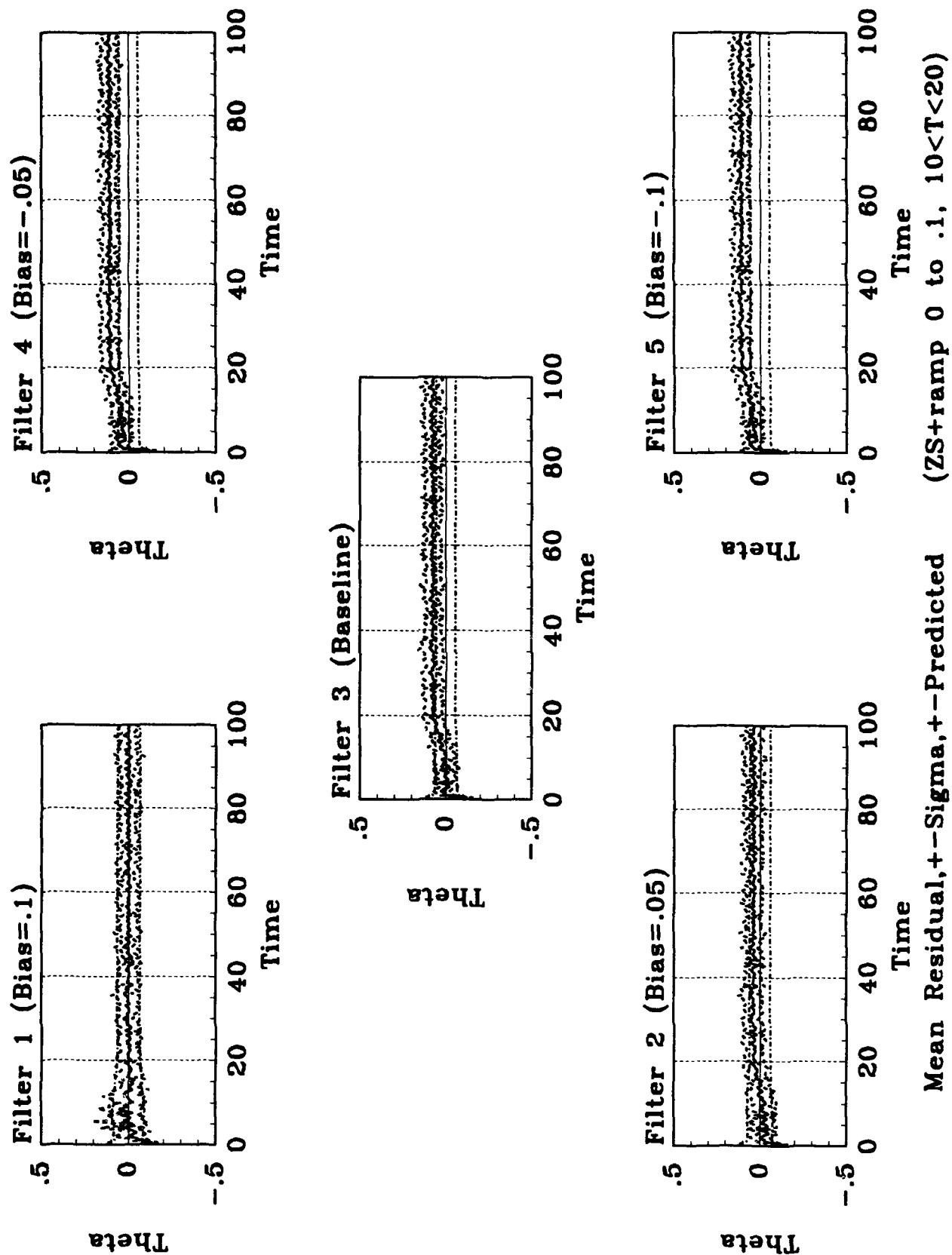


Figure B.35. Case #5 Theta Residuals

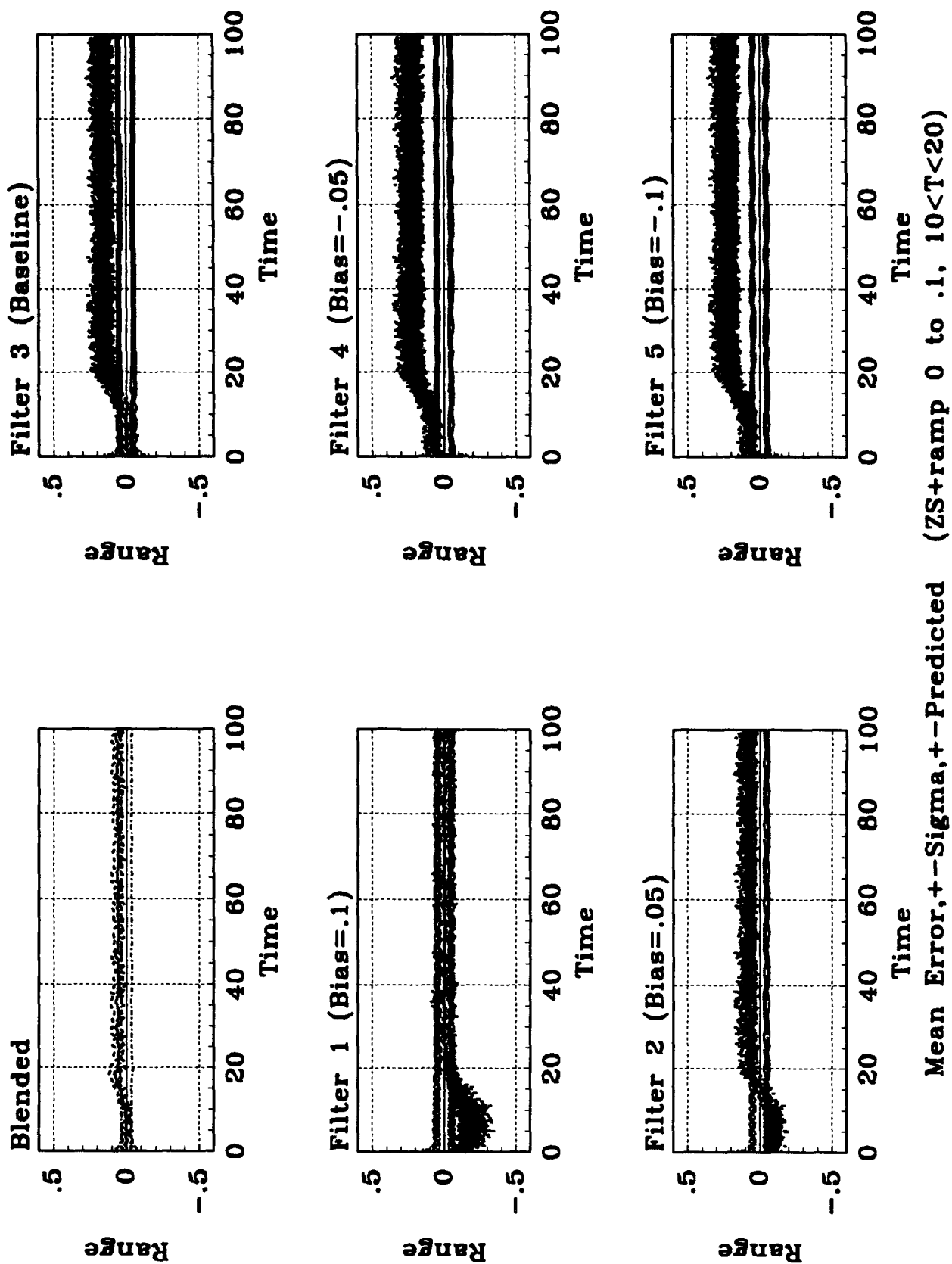


Figure B.36. Case #5 Range
B-42

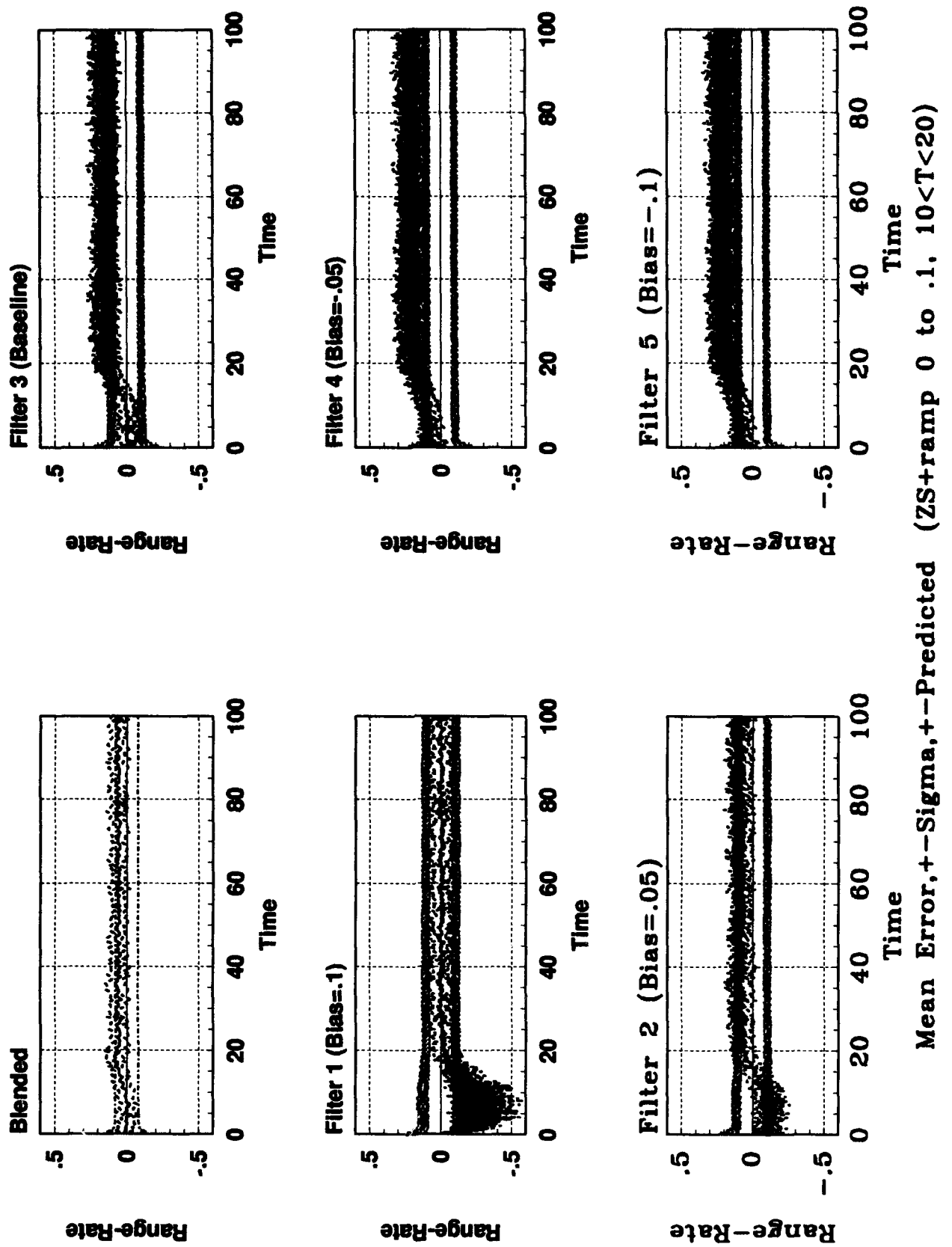


Figure B.37. Case #5 Range-rate
B-43

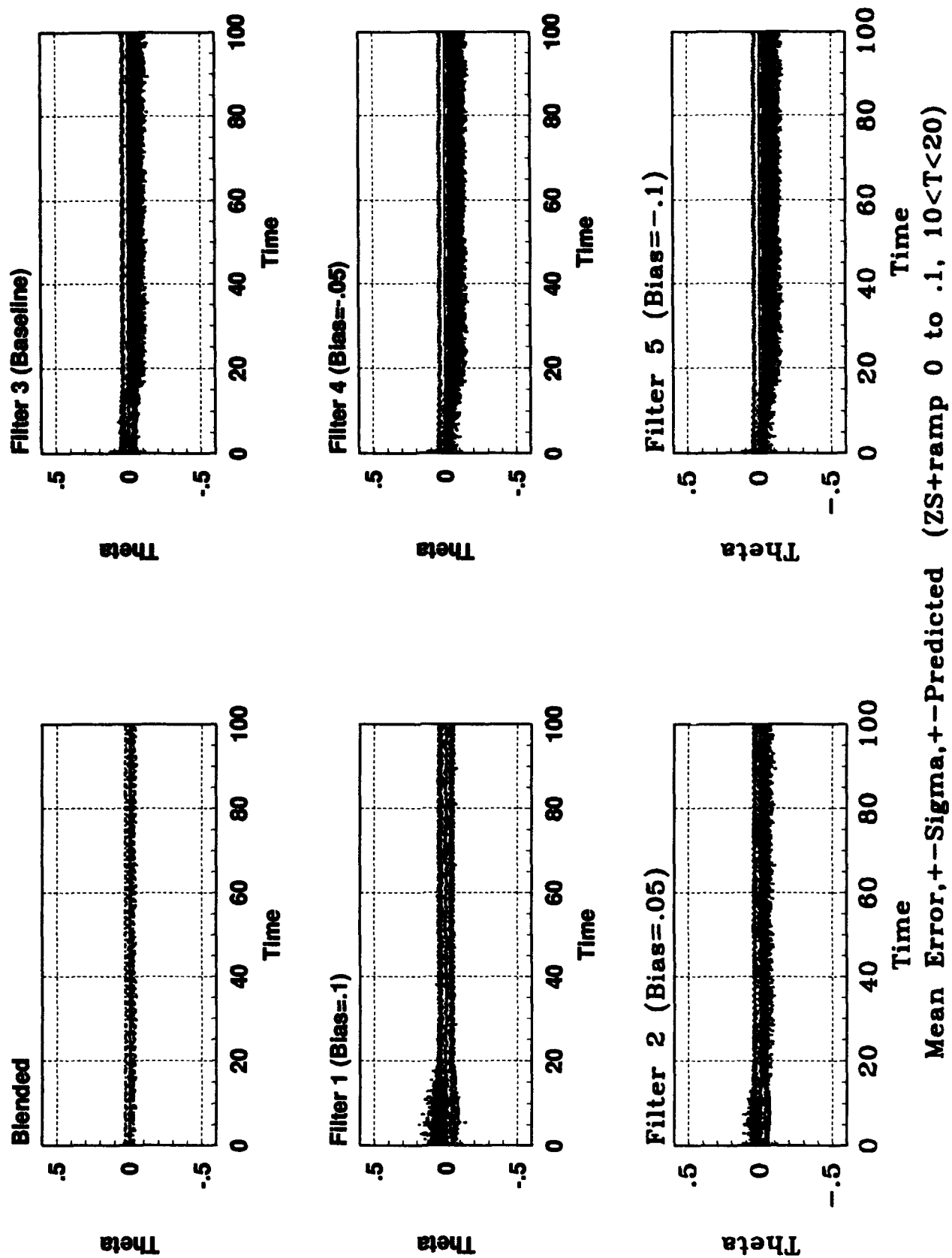


Figure B.38. Case #5 Theta
B-44

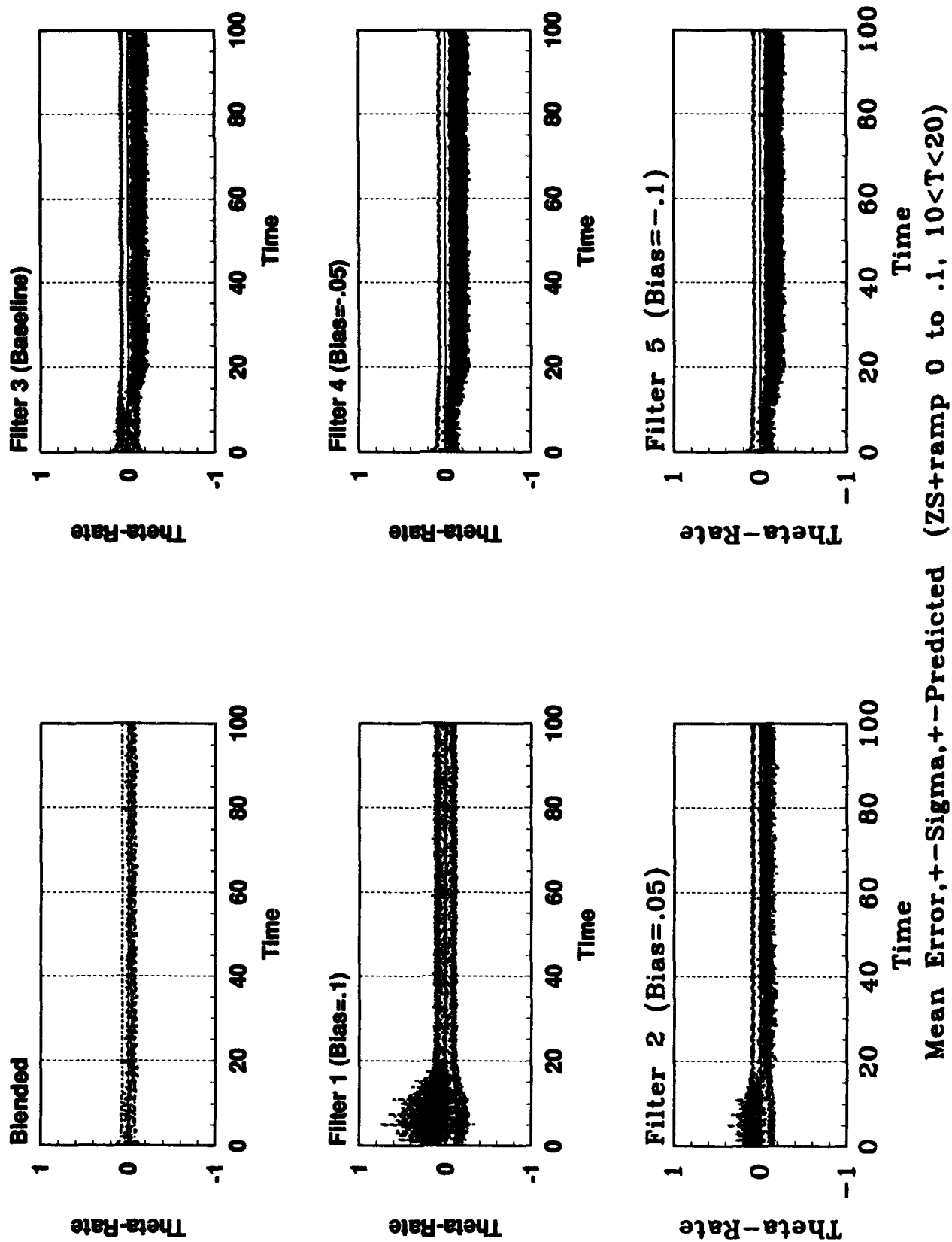


Figure B.39. Case #5 Theta-rate
B-45

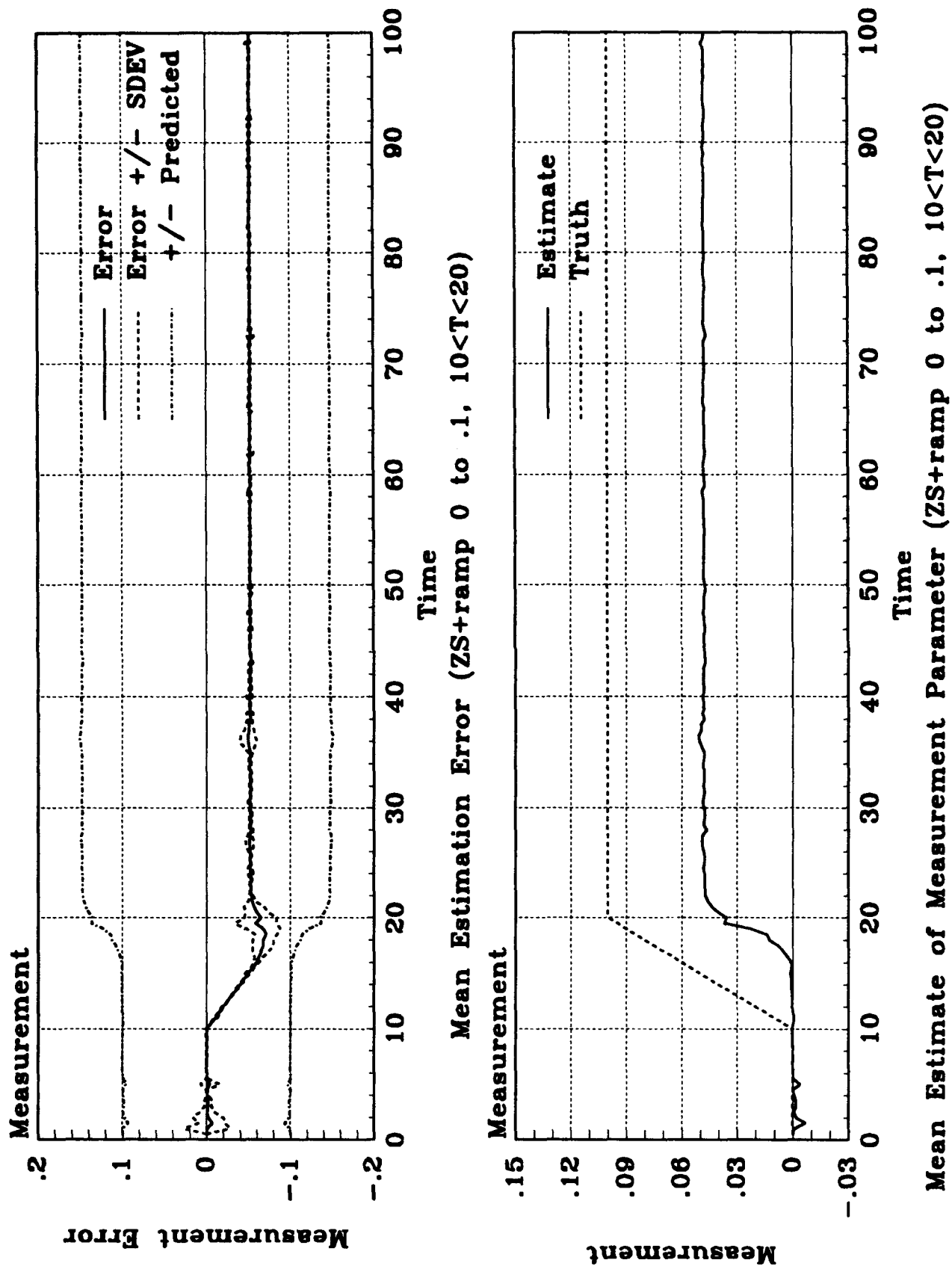


Figure B.40. Case #5 Parameter

B.6 Measurement Bias to .15 for $T > 10$

Table B.6. Orbit Problem Failure Case #6

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
6	Step in Measurement	$Z_S = Z_{S0} + .15Z_{S0}$	$T > 10$	$Z_J + BIAS_{Z_J}$

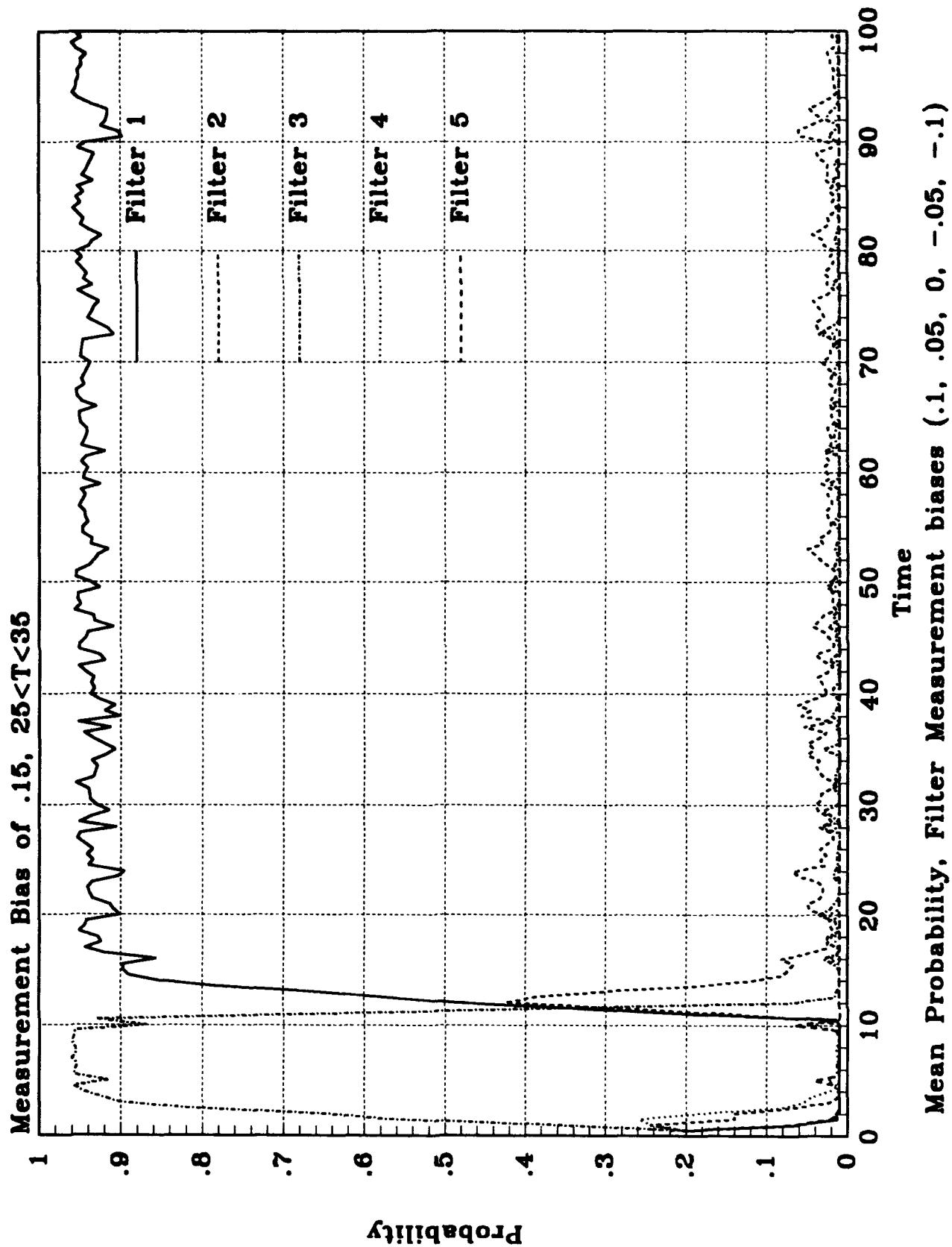


Figure B.41. Case #6 Probabilities

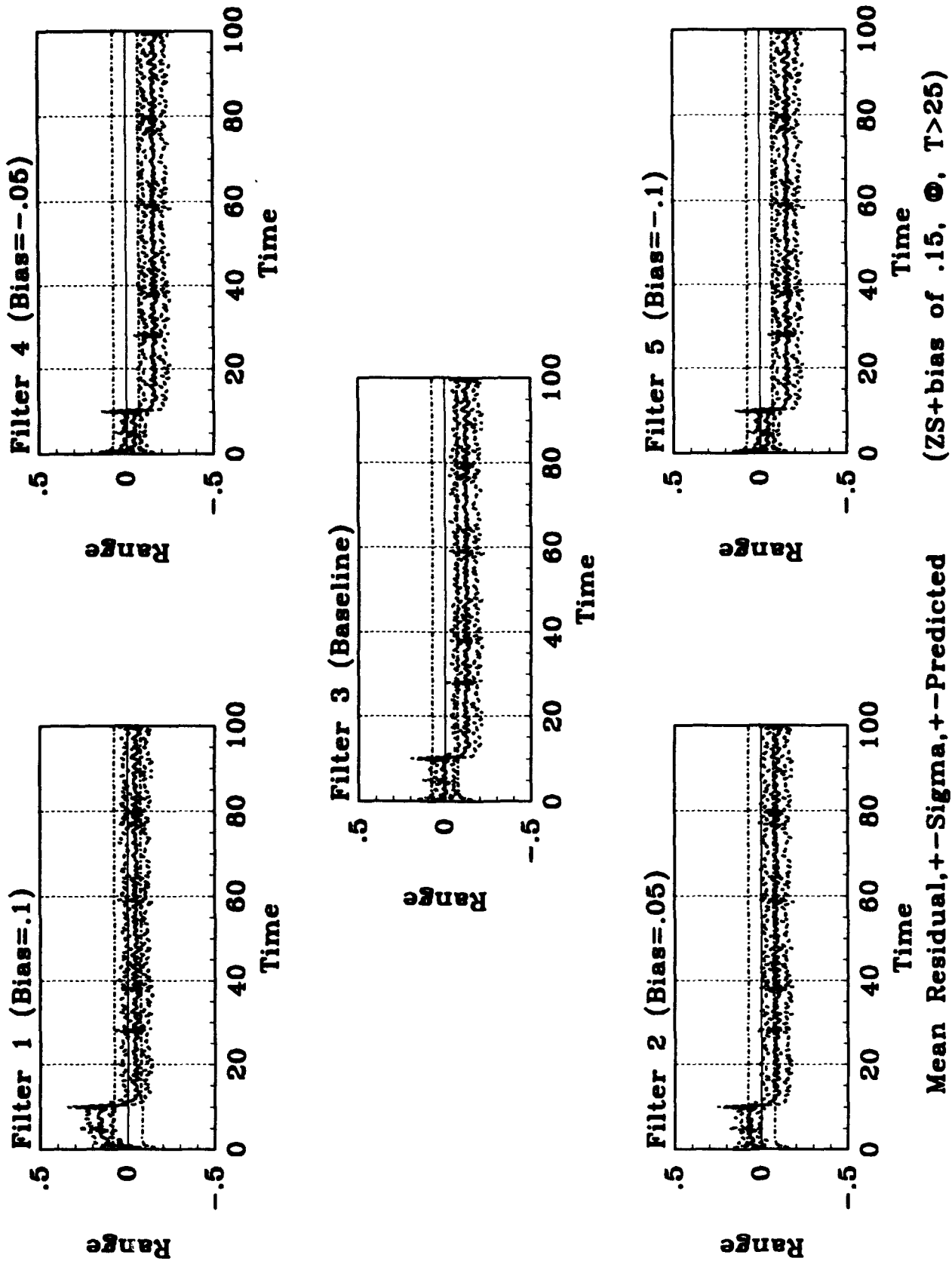


Figure B.42. Case #6 Range Residuals

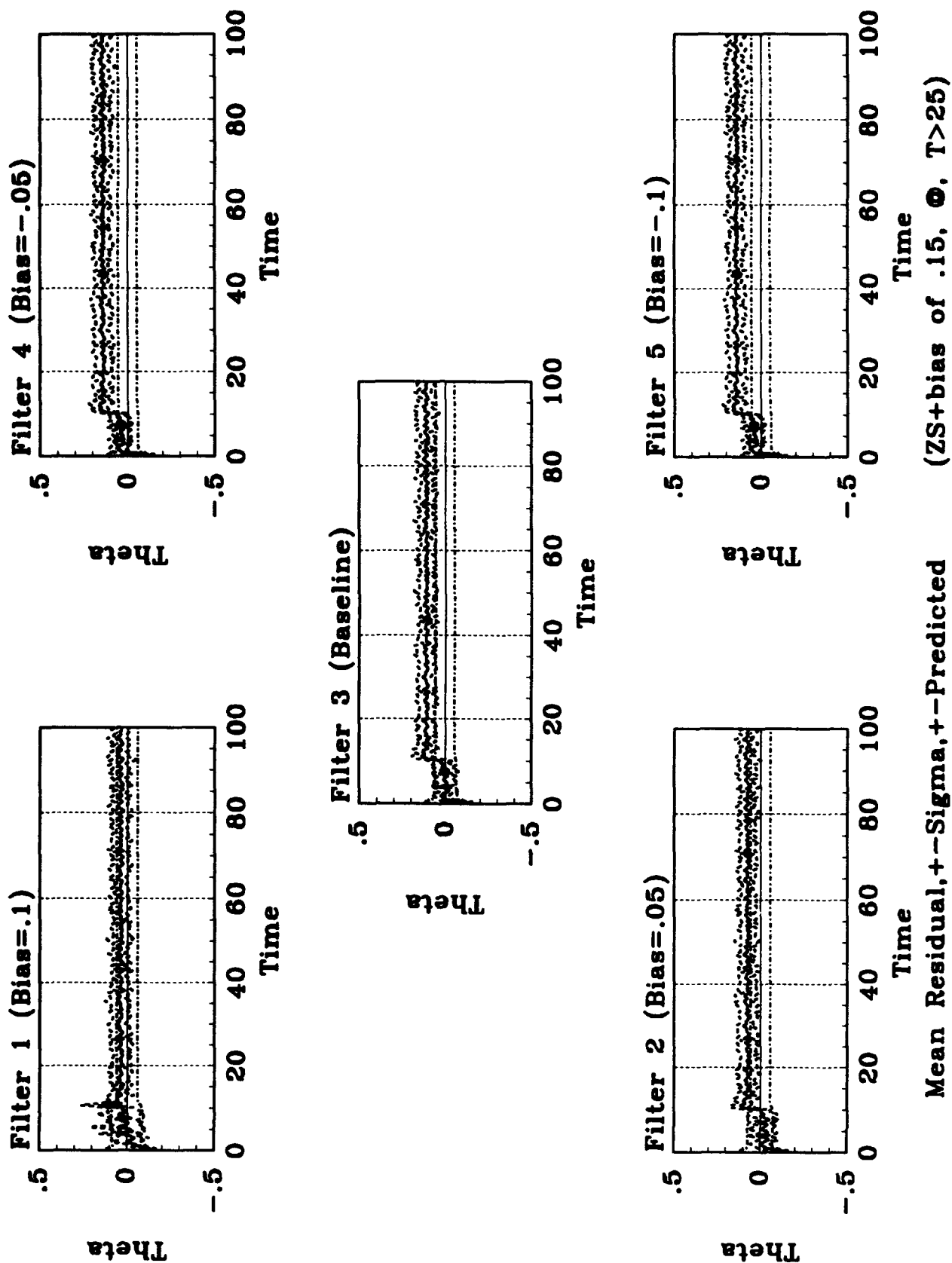
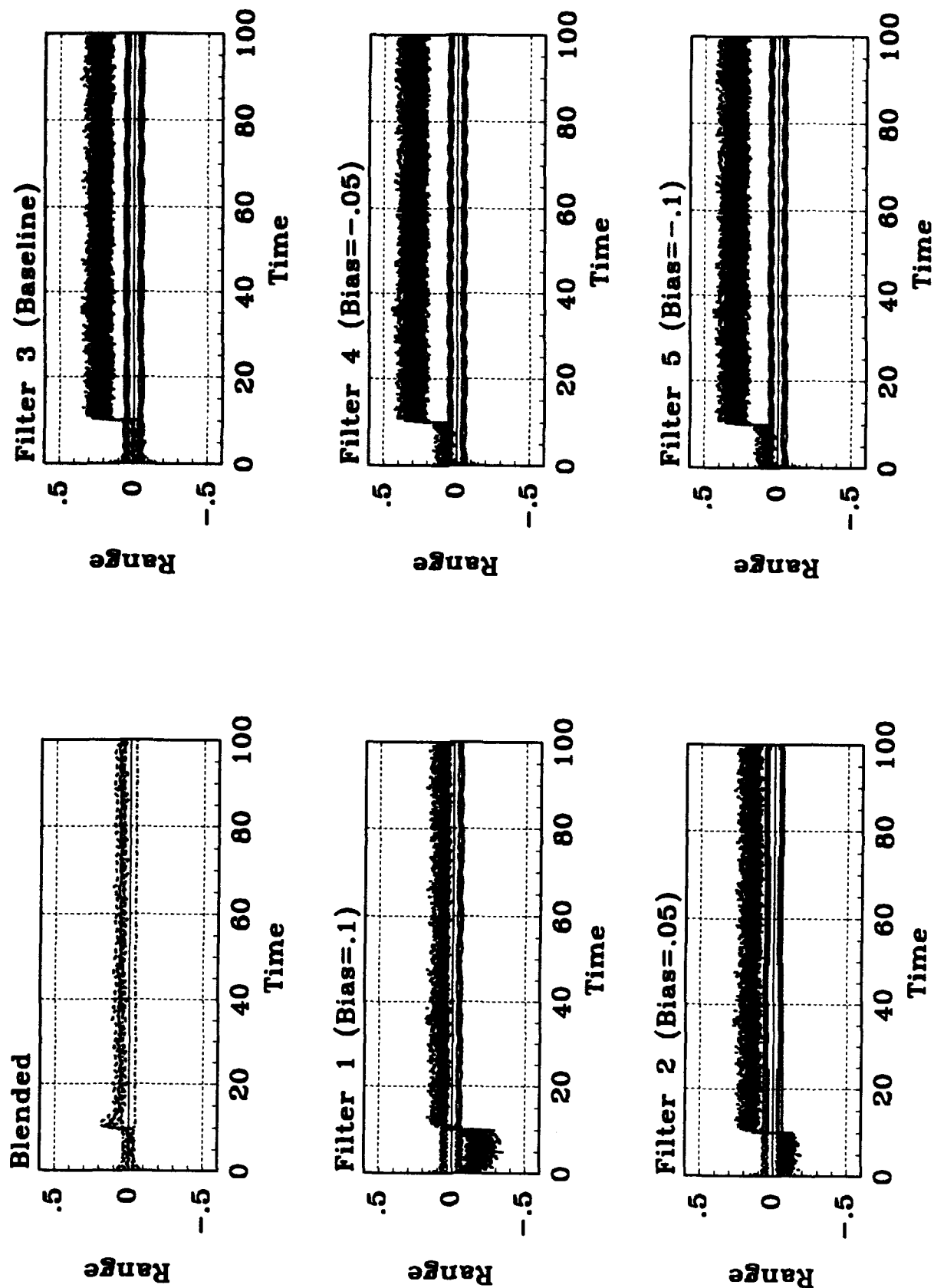


Figure B.43. Case #6 Theta Residuals



Mean Error, +-Sigma, +-Predicted (ZS+bias of .15, @, T>25)

Figure B.44. Case #6 Range
B-51

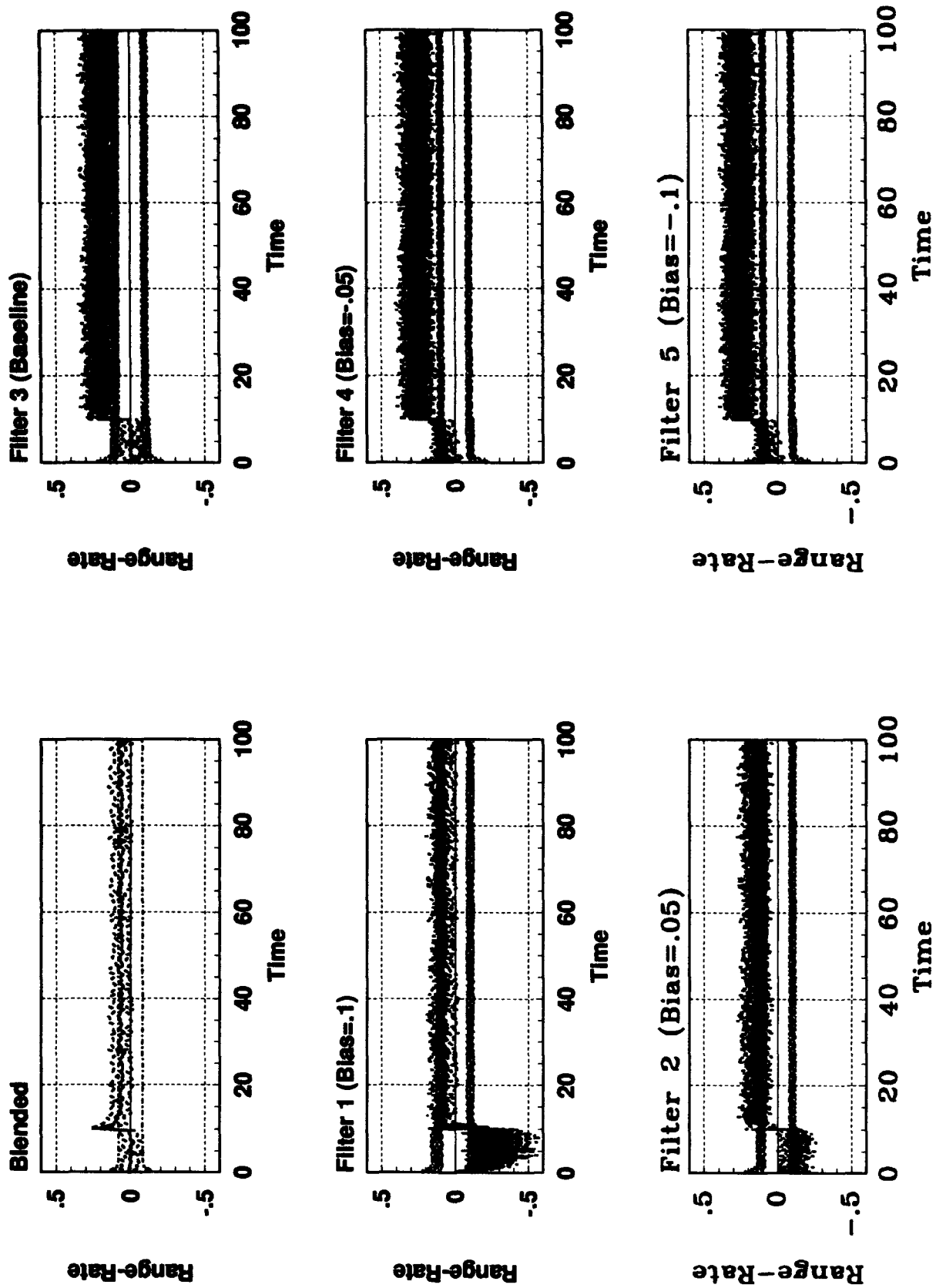


Figure B.45. Case #6 Range-rate
B-52

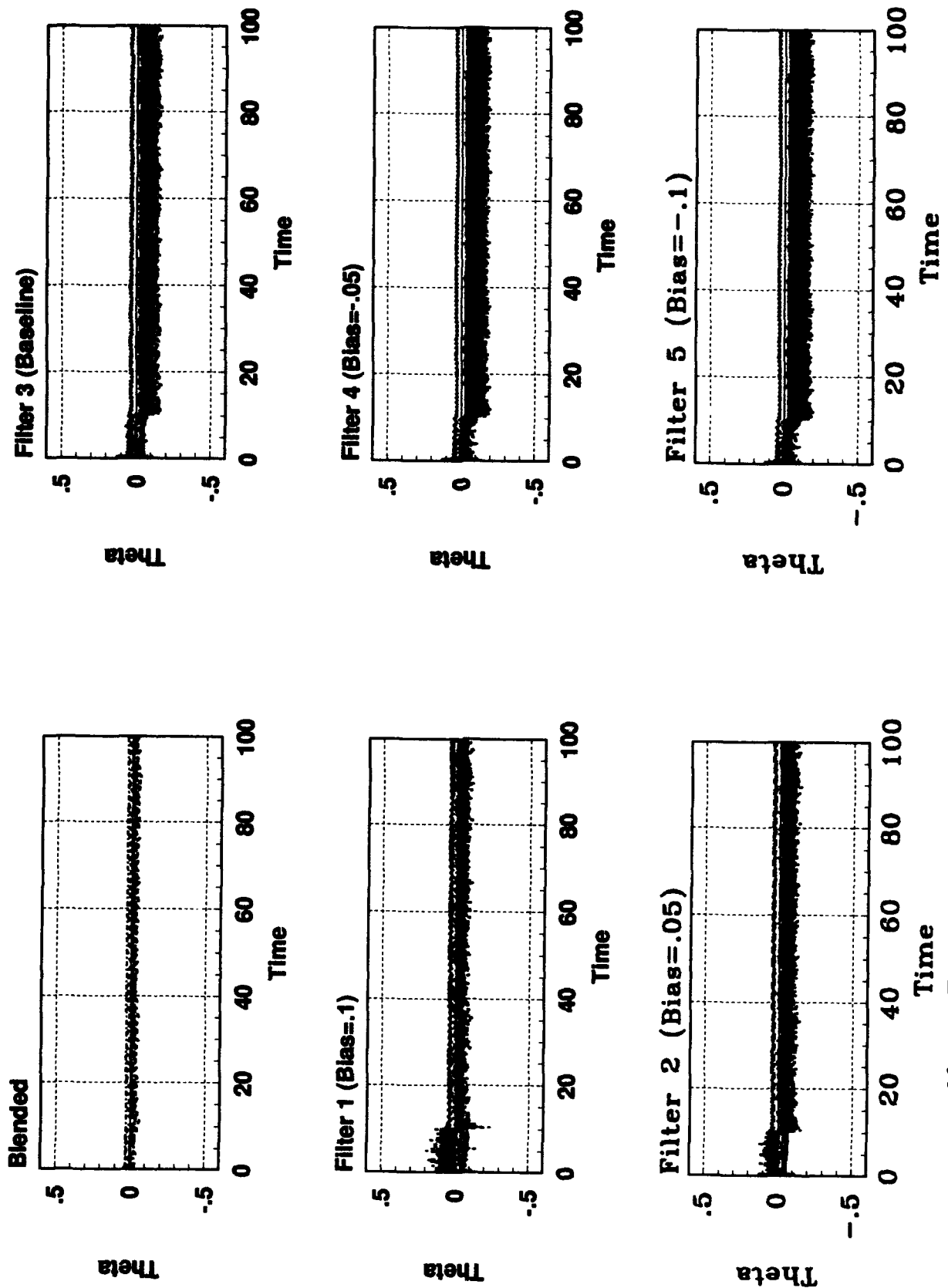
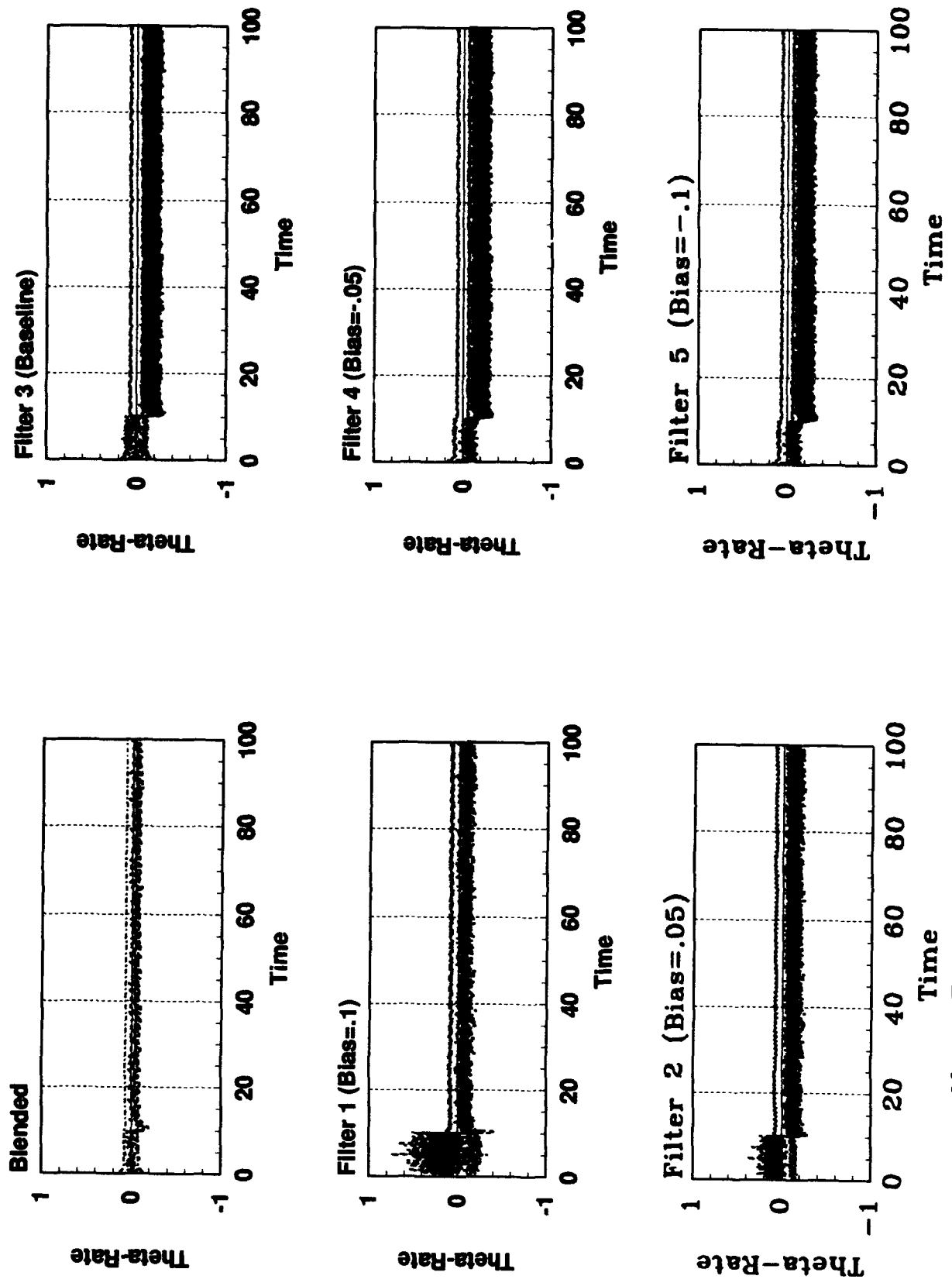


Figure B.46. Case #6 Theta



Mean Error, +-Sigma, +-Predicted (ZS+bias of .15, Θ , $T > 25$)

Figure B.47. Case #6 Theta-rate
B-54

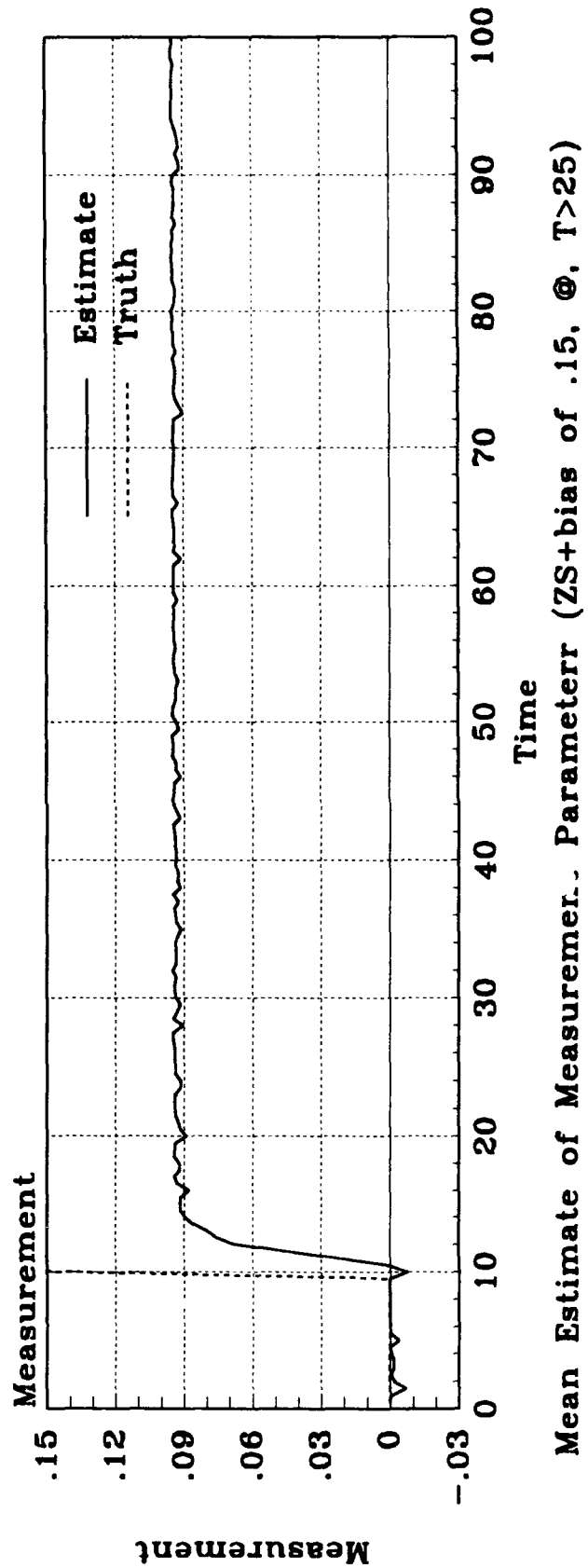
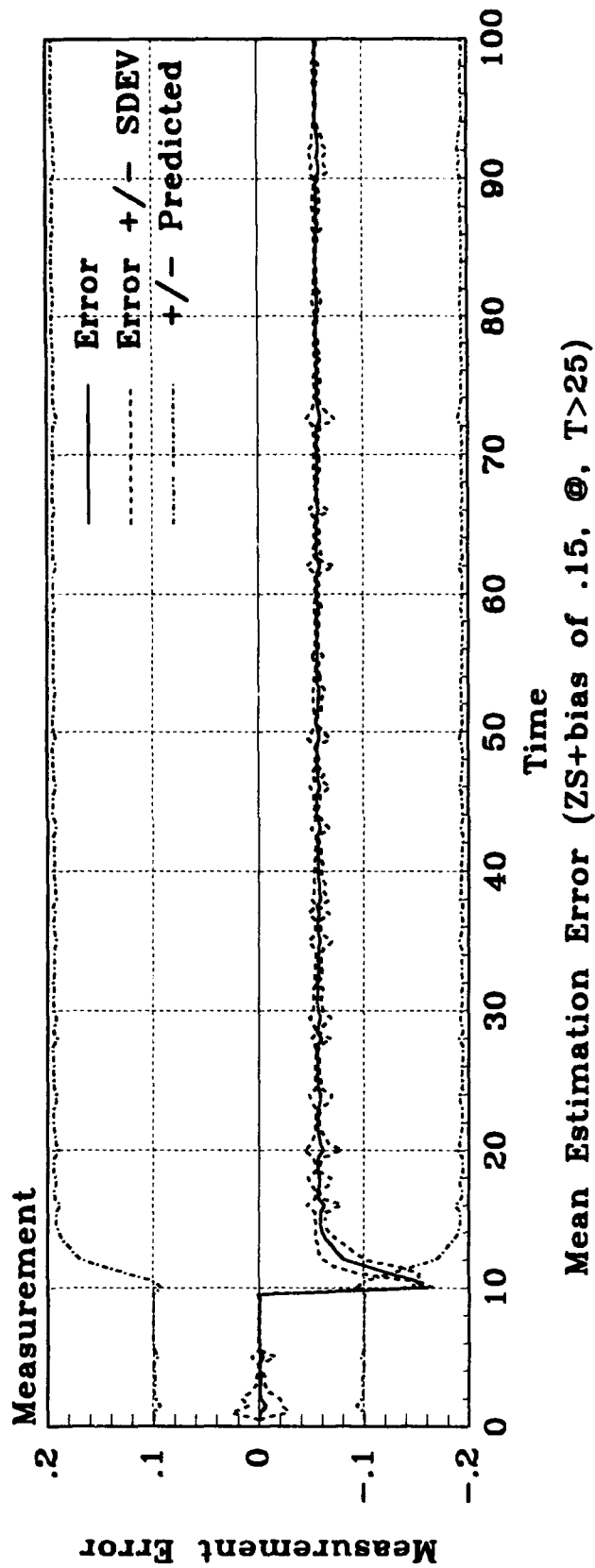


Figure B.48. Case #6 Parameter

B.7 Measurement Ramp to .15 for $10 < T < 25$

Table B.7. Orbit Problem Failure Case #7

Error Case #	Failure Type In Truth Model	Failure Induced	Failure Time	Elemental Filter
7	Ramp in Measurement	$Z_S = Z_{S0} + \frac{.15Z_{S0}(T-10)}{(15)}$ $Z_S = Z_{S0} + .15Z_{S0}$	$10 < T < 25$ $T > 25$	$Z_I + BIAS_z,$

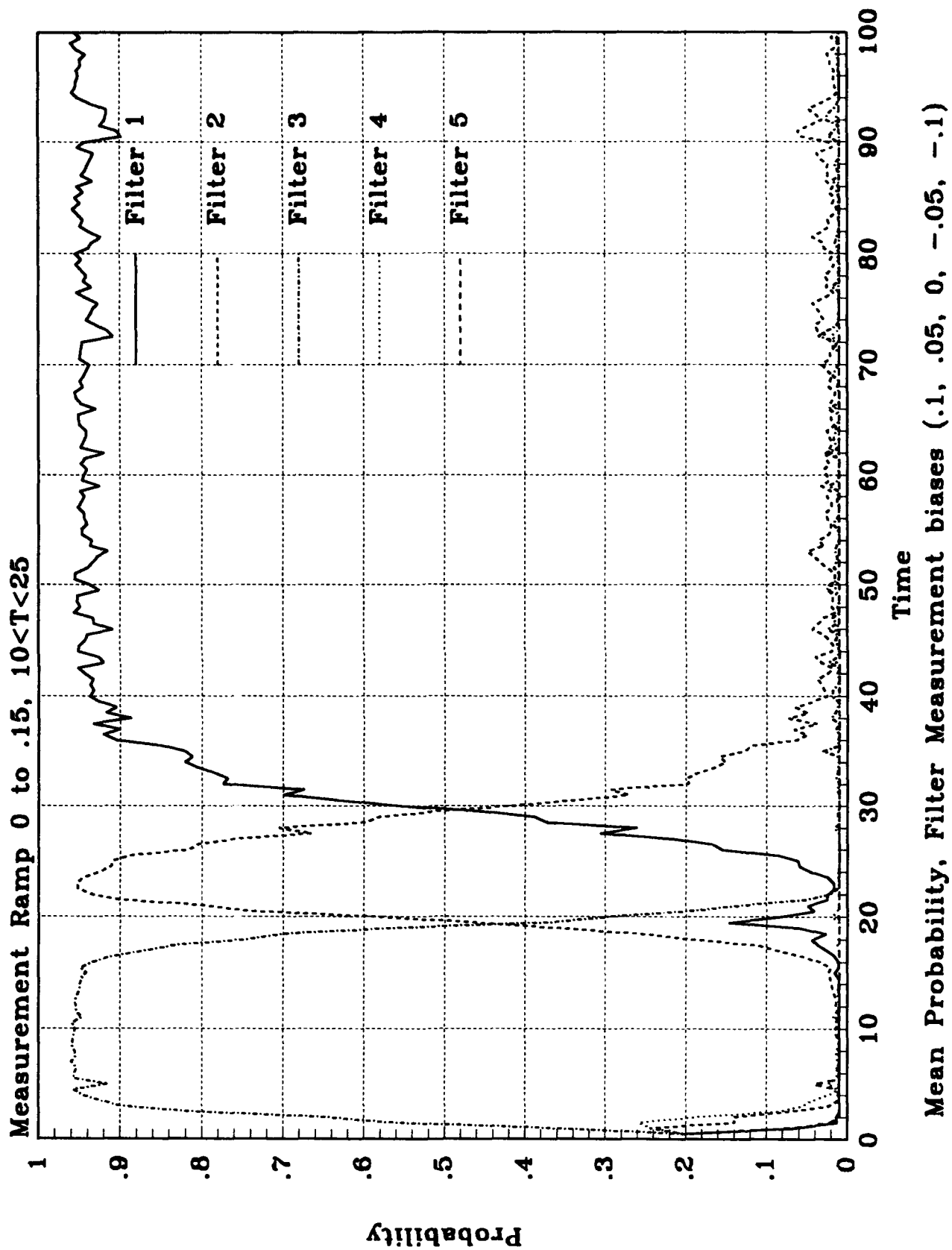


Figure B.49. Case #7 Probabilities
B-57

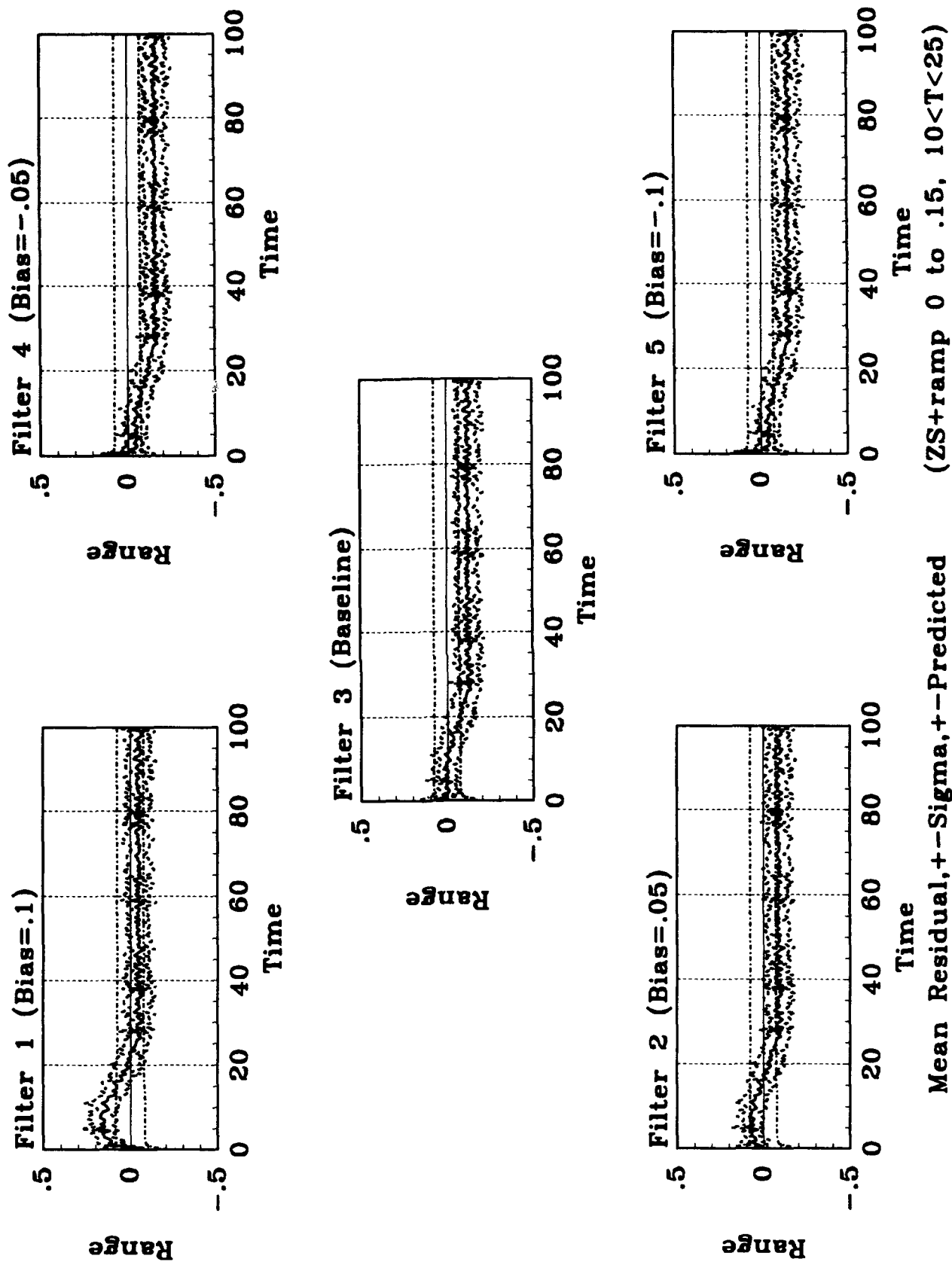


Figure B.50. Case #7 Range Residuals

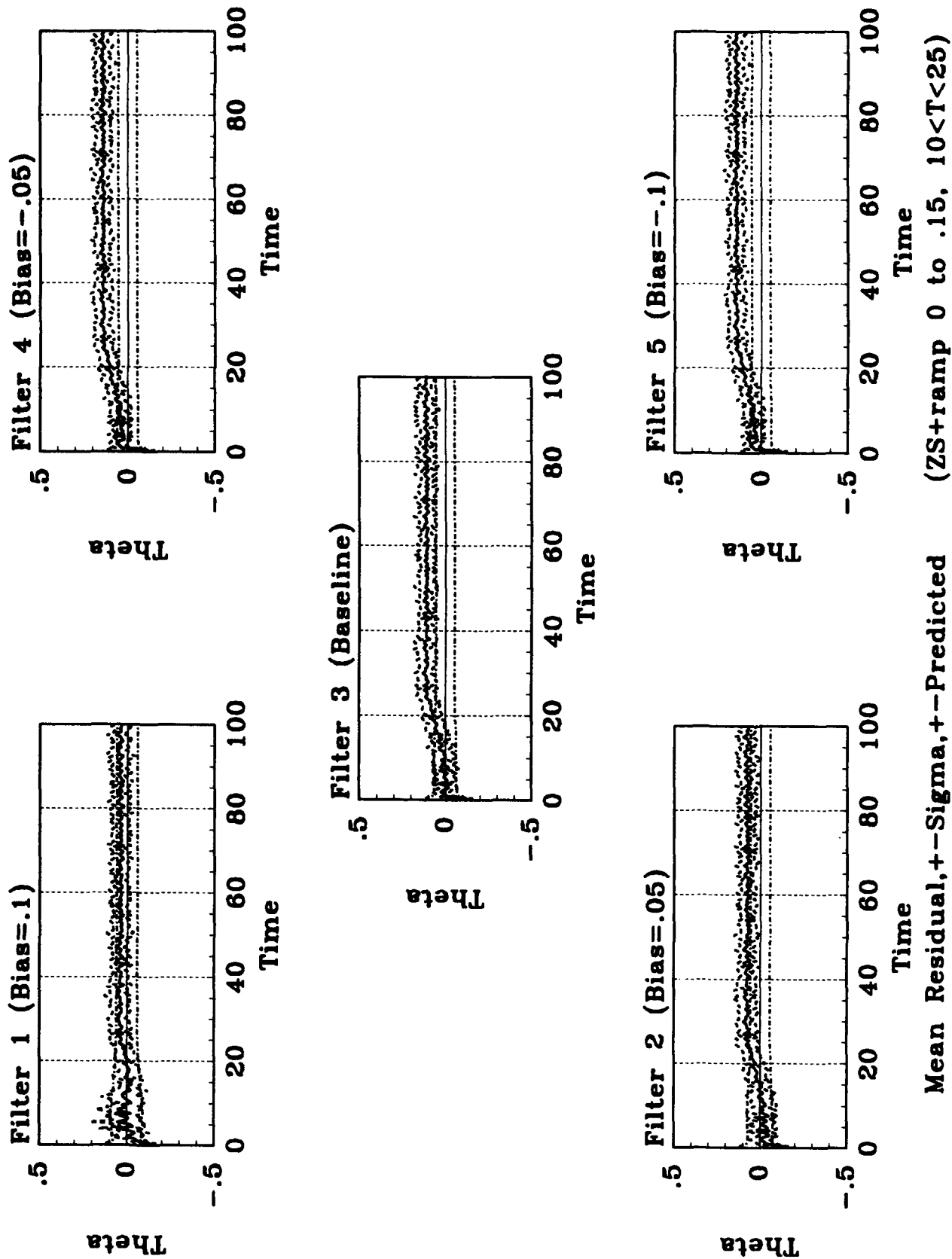


Figure B.51. Case #7 Theta Residuals

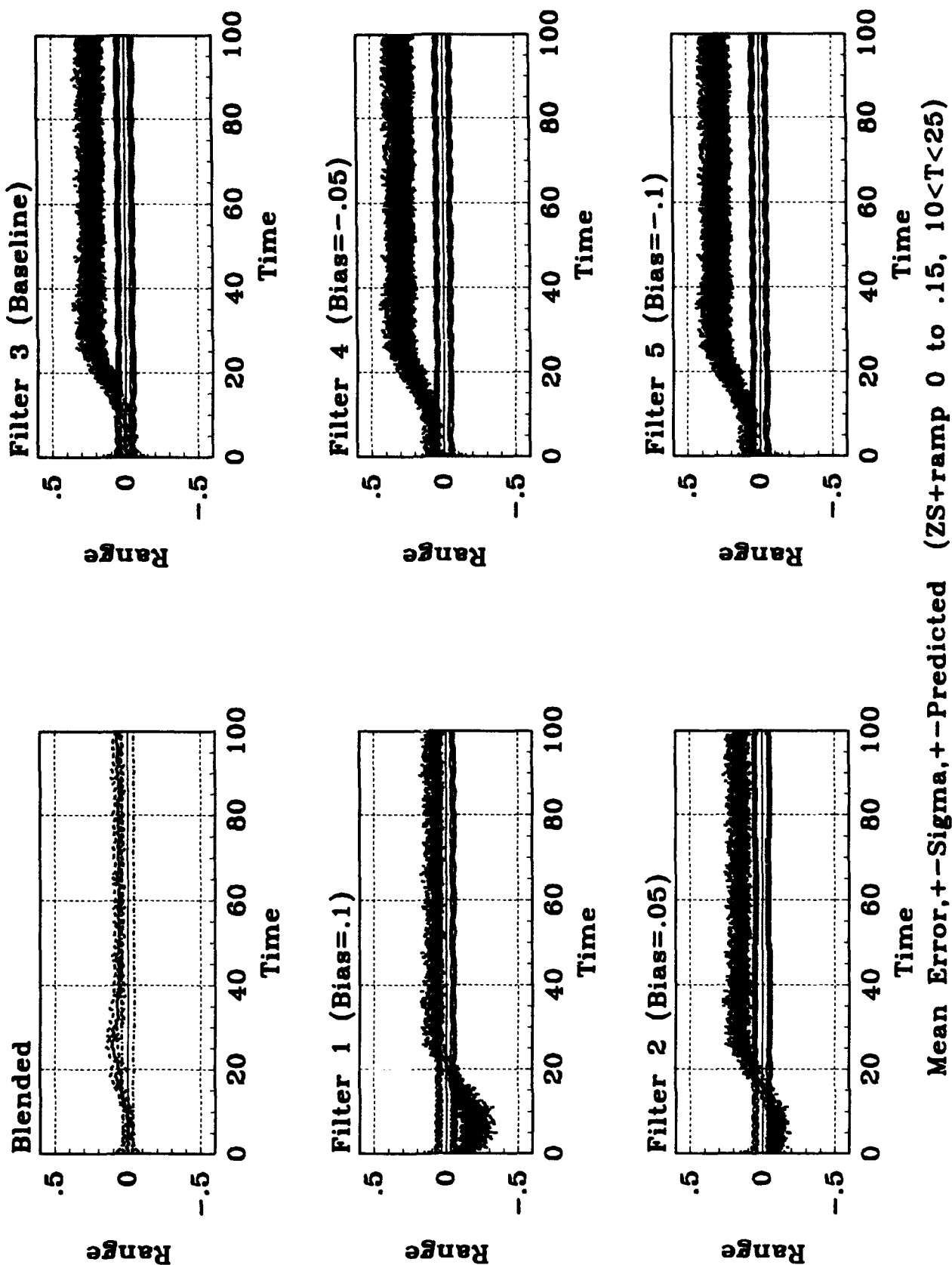


Figure B.52. Case #7 Range

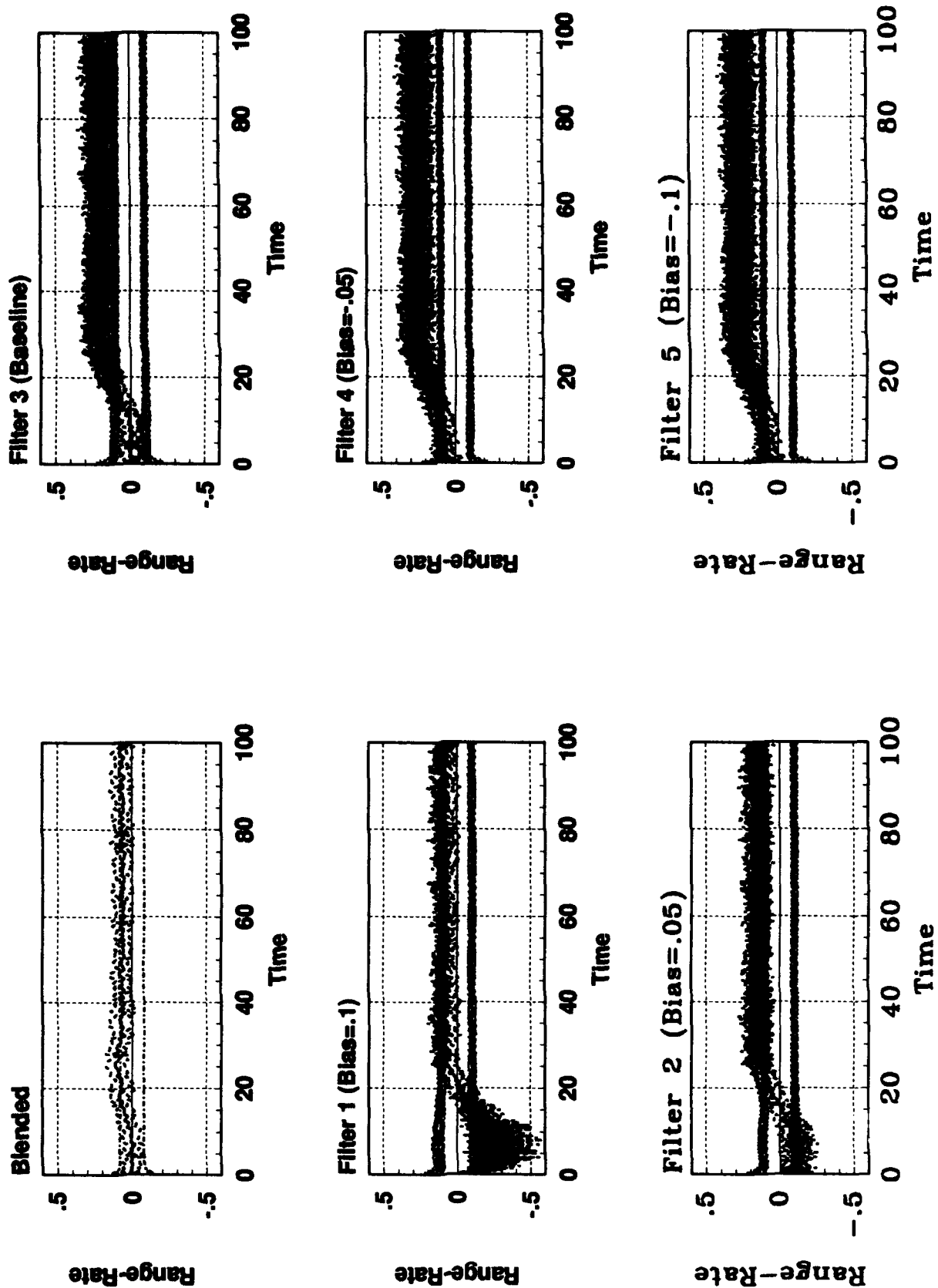
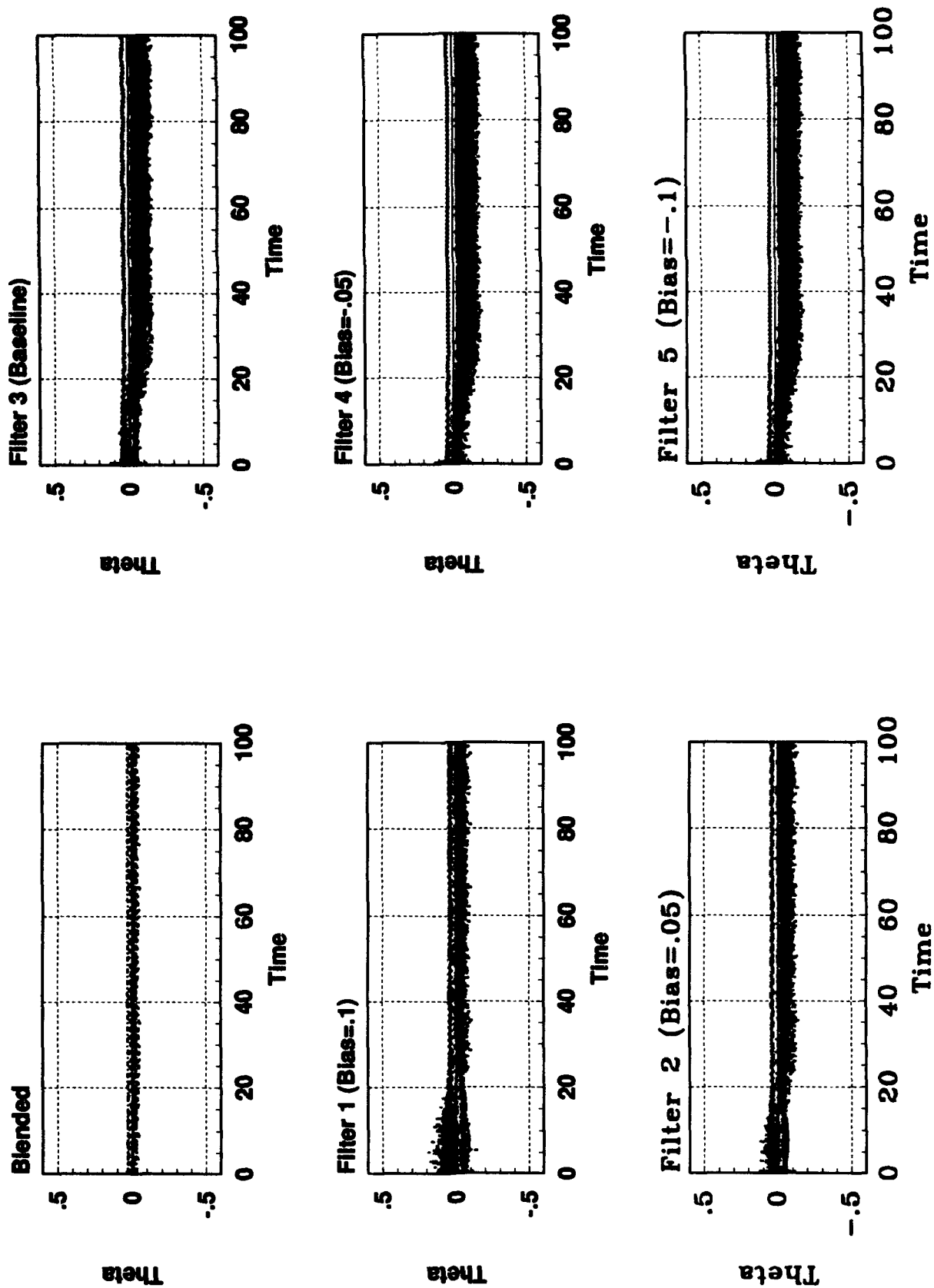


Figure B.53. Case #7 Range-rate
B-61



Mean Error, $+-\text{Sigma}$, $+-\text{Predicted}$ ($ZS+\text{ramp}$ 0 to .15, $10 < T < 25$)

Figure B.54. Case #7 Theta

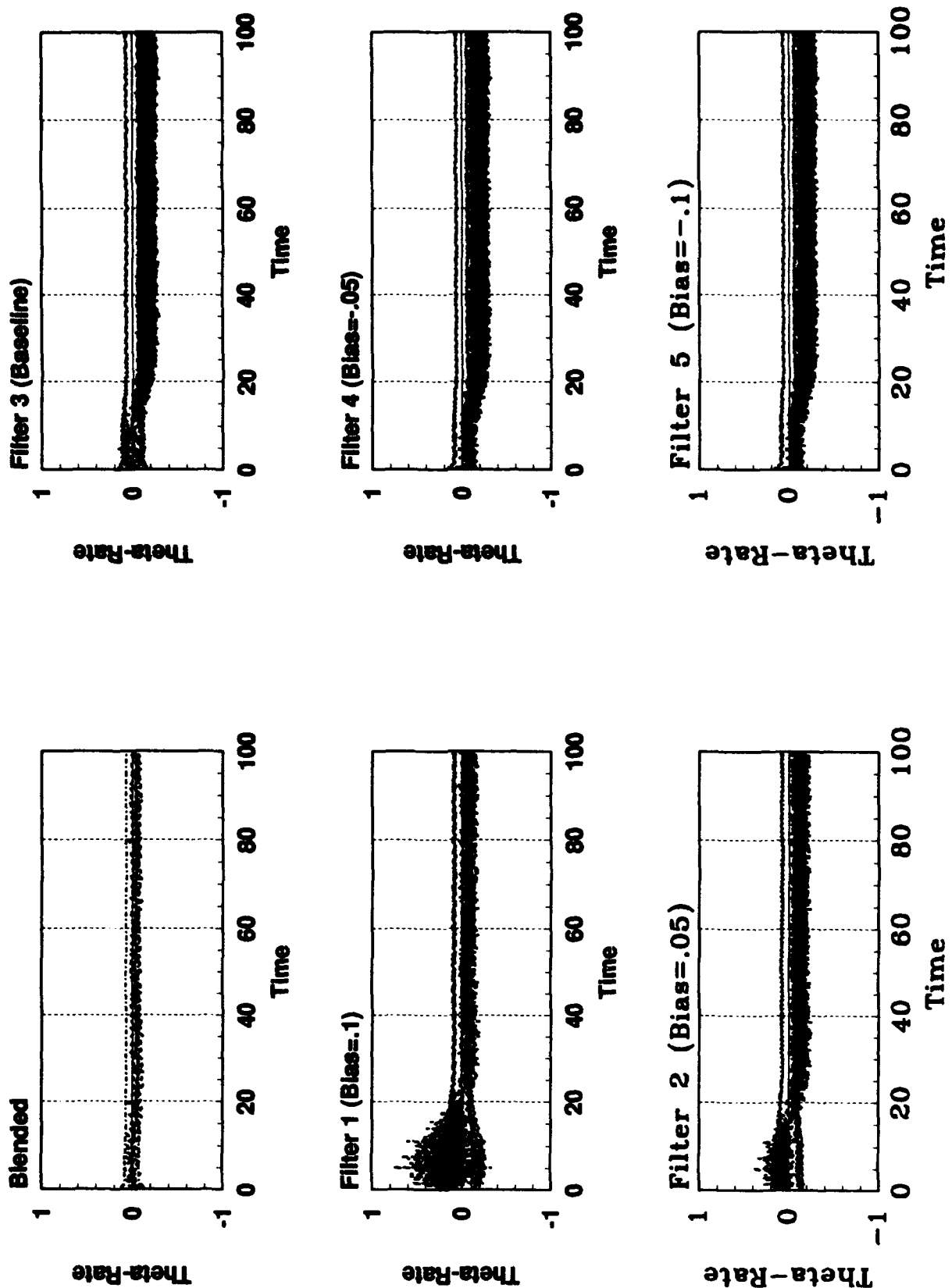


Figure B.55. Case #7 Theta-rate
B-63

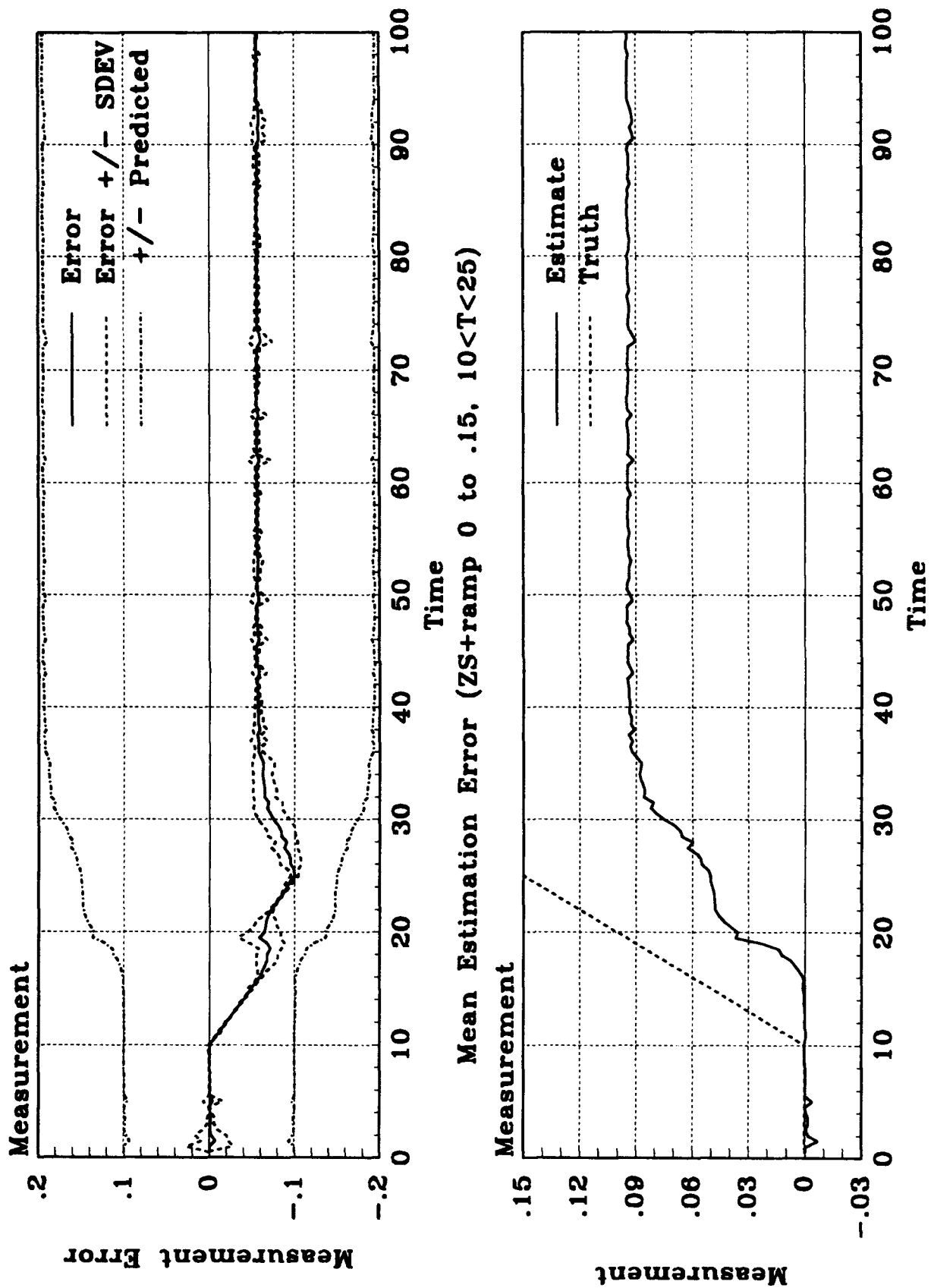


Figure B.56. Case #7 Parameter
 B-64

Bibliography

1. Britting, Kenneth R. *Inertial Navigation Systems Analysis*. Wiley-Interscience, 1971.
2. Brown, R. and P. Hwang. "GPS Failure Detection by Autonomous Means within the Cockpit," *Institute of Navigation* (1986).
3. Carlson, Neal A. *DKFSIM 1.1 User's Manual*. Contract No. F33615-87-C-1520, Document No. TM-89-005.. Technical Report, Integrity Systems, Inc., Winchester, MA, December 1989.
4. Carlson, Neal A., et al. *Program Design Description for a Multimode Simulation for Optimal Filter Evaluation (MSOFE)*. Technical Report, Integrity Systems, Inc., Winchester, MA and Air Force Avionics Laboratory, WPAFB, Ohio, February 1989. AFWAL-TR-88-1137.
5. Ching, M., et al. *Performance of an Integrated INS and GPS with Jamming Detection*. Technical Report, Air Force Institute of Technology, WPAFB, Ohio, Spring 1992.
6. Desai, M., et al. "Dual Sensor Failure Identification Using Analytic Redundancy," *AGARD Lecture Series No. 82* (1976).
7. Gustafson, John A. *Control of a Large Flexible Space Structure using Multiple Model Adaptive Estimation Algorithms*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1991.
8. Hanlon, Peter D. *Failure Identification Using Multiple Model Adaptive Estimation for the LAMBDA Flight Vehicle*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1992.
9. Herrera, Theodore D. *Kalman Filter Tracking of a Reflective Target Using Forward Looking Infrared Measurements and Doppler Returns*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1991.
10. Integrated Systems Inc., Santa Clara, CA. Matrix_x, August 1988. Version 1.2/1.3.
11. Jr., Robert W. Lashlee. *Moving-Bank Multiple Model Adaptive Estimation Applied to Flexible Spacestructure Control*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1989.
12. Knudsen, L. *Performance Accuracy (Truth Model/Error Budget) Analysis for the LN-93 Inertial Navigation System Inertial Navigation Unit*. Technical Report, 5500 Canoga Avenue, Woodland Hills, California 91365: Litton Guidance and Control Systems, January 1985. DID No. DI-S-21433 B/T: CDRL No. 1002.
13. Korn, J. and L. Beean. *Application of Multiple Model Adaptive Estimation Algorithms to Maneuver Detection and Estimation*. Technical Report, Alphatech Inc., Burlington, MA, June 1993.
14. Lainiotis, D. G. and S. D. Likothanasis. "Adaptive Control Algorithms - A Comparative Computational Analysis-Parallelism," *Control and Computers*, ISSN 0315 8934, vol. 16, 22-27 (1988).
15. Lewantowicz, Z. H. and D. W. Keen. "Graceful Degradation of GPS/INS Performance With Fewer Than Four Satellites," *The Institute of Navigation, National Technical Meeting*, 269-275 (Jan 1991).

16. Lin, Ching-Fang. *Modern Navigation, Guidance, and Control Processing*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.
17. Martin, E. H. "GPS User Equipment Error Models," *The Institute of Navigation, volume I*:109-118 (1980).
18. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, I*. Academic Press, Inc., New York, NY, 1979.
19. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, II*. Academic Press, Inc., New York, NY, 1982.
20. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, III*. Academic Press, Inc., New York, NY, 1982.
21. Menke, Timothy E. *Multiple Model Adaptive Estimation Applied to the VISTA F-16 with Actuator and Sensor Failures*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, June 1992.
22. Musick, Stanton H. *PROFGEN - A Computer Program for Generating Flight Profiles*. Technical Report, Air Force Avionics Laboratory, WPAFB, Ohio, November 1976. AFAL-TR-76-247, DTIC ADA034993.
23. Musick, Stanton H. and Neal A. Carlson. *MISOFE - Multimode Simulation for Optimal Filter Evaluation*. Technical Report, Air Force Avionics Laboratory, WPAFB, Ohio, October 1980. AFWAL-TR-88-1136.
24. Musick, Stanton H. and Neal A. Carlson. *Users' Manual for a Multimode Simulation for Optimal Filter Evaluation (MISOFE)*. Technical Report, Air Force Avionics Laboratory, WPAFB, Ohio, and Integrity Systems, Inc., Winchester, MA, April 1990. AFAL-TR-88-1138.
25. Negast, William Joseph. *Incorporation of Differential Global Positioning System Measurements Using an Extended Kalman Filter for Improved Reference System Performance*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1991.
26. Niblett, Barbara J. *A Multiple Model Adaptive Estimator Using First and Second-Order Acceleration Models for Use in a Forward-Looking Infrared Tracker*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, June 1991.
27. Ritland, John T. "Impact of Inertial System Quality on GPS-Inertial Performance in a Jamming Environment," *American Institute of Aeronautics and Astronautics*, 1459-1467 (1987).
28. Robert Nelson Riggins, Jr. *Detection and Isolation of Plant Failures in Dynamic Systems*. PhD dissertation, University of Michigan, 1991.
29. Sablan, Samuel J. *Multiple Model Adaptive Estimation Techniques for Adaptive Model-Based Robot Control*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1989.
30. Schore, Michael R. *Robustness of a Moving-Bank Multiple Model Adaptive Estimation Applied to Large Space Structure*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1989.
31. Snodgrass, Faron Britt. *Continued Development and Analysis of a New Extended Kalman Filter for the Completely Integrated Reference Instrumentation System (CIRIS)*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, March 1990.

32. Stacey, Richard D. *A Navigation Reference System Using Global Positioning System and Transponder Aiding*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, March 1991.
33. Stevens, Richard D. *Reconfigurable Flight Control via Multiple Model Adaptive Control Methods*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1990.
34. Theory and Applications of Kalman-Filtering, AGARDograph 139. *Chapter 10 - General Questions on Kalman Filtering in Navigation Systems*, NATO/AGARD, London, England.
35. Van Trees, H. L. *Detection, Estimation and Modulation Theory*. Wiley and Sons, New York, New York, 1968.
36. Vasquez, Juan R. *Detection of Spoofing, Jamming, or Failure of a Global Positioning System (GPS)*. MS thesis, Air Force Institute of Technology, WPAFB, Ohio, December 1992.
37. Willsky, A. and H. Jones. "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes," *The Analytic Sciences Corporation, Reading, Mass., under USAF Contract No. F04701-74-C-0095* (1976).
38. Willsky, Alan S. "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, 601-611 (1976).

Vita

Capt Robert L. Nielsen was born in LaSalle, IL, on 15 Jan 48. Growing up on a farm, he graduated from Lostant Community High School, Lostant, IL in 1966. He attended Bradley University, Peoria, IL and Illinois Valley Community College, Peru, IL before enlisting in the US Army in 1968. He was a Counter Intelligence Agent in the US Army, serving in West Germany. After separating from the US Army in 1971, he stayed in West Germany for several more years where he owned and managed a Gastehaus in Bad Kohlgrub, Bavaria. Returning to the United States in 1973, he returned to college and graduated from Northern Illinois University, Dekalb, IL with a BS in Physics in 1977. While in college, he met Vera J. Williams whom he married in 1974. After college, he attended graduate school until 1980 at Denver Baptist Seminary, Denver, CO, where he studied theology, Bible, and counseling. While and after attending seminary (1977 - 1982), he owned and operated a painting and repair business, was a licensed commercial real estate broker, property manager and small business consultant. While living in Denver, he and Vera had two children, Lynsa and Eric. He joined the US Air Force in 1982 to attend the Air Force Institute of Technology (AFIT), Wright Patterson AFB (WPAFB), OH, graduating with a BS in Aeronautical Engineering in 1984. He then worked as a Flight Test Engineer, Guidance System Analyst, and was Chief of the Guidance Analysis Section in the Guidance Division, 6585th Test Group, Holloman AFB, NM, until 1989. While living in Alamogordo, NM, Bob and Vera had two more children, Amber and Lauren. Returning to WPAFB, he was Deputy Chief Flight Systems Engineer in the Short Range Attack Missile System Program Office, Aeronautical Systems Division, until again returning to AFIT in January, 1992. This time at AFIT, he earned a MS in Electrical Engineering in Guidance and Control in December 1993, and was assigned to the Avionics Directorate, Aeronautical Systems Center, WPAFB, OH.

Permanent address: 2726 Greene Hills Dr.
Beavercreek, OH 45324

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

December 1993

3. REPORT TYPE AND PERIODICITY

Master's Thesis

4. TITLE AND SUBTITLE

Development of a Performance Evaluation
Tool (MMSOFE) for Detection of Failures
with Multiple Model Adaptive Estimation (MMAE)

6. AUTHOR(S)

Robert L. Nielsen
Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFTT/GE/ENG/93S-37

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Enter number of words)

Abstract

Multiple model Kalman Filter (KF) techniques are used extensively for Multiple Model Adaptive Estimation (MMAE), Multiple Model Adaptive Control (MMAC), and Distributed Kalman Filter (DKF) applications to determine Bayesian-blended optimal estimates of states, uncertain parameters, and optimal control signals. Multiple model methods are used for sensor management, Failure Detection and Isolation (FDI), and other Guidance and Control (G&C) applications. A simulation tool called the Multiple Model Simulation for Optimal Filter Evaluation (MMSOFE) has been in this research. MMSOFE is based on the well-benchmarked single Kalman filter tool called Multimode Simulation for Optimal Filter Evaluation (MSOFE). MMSOFE is a highly portable and versatile multiple and single Kalman filter evaluation tool. It is capable of performing simulations with one filter or up to 98 elemental filters in a multiple model adaptive filter structure. It can be adapted easily for other multiple model applications. MMSOFE was applied to failure detection and isolation of measurement jamming- and spoofing-type failures, similar to jamming and spoofing of a Global Positioning System (GPS). A satellite orbit estimation test case was used.

14. SUBJECT TERMS

Multiple Model, MMSOFE, MSOFE, Kalman Filter, Bayesian, Multiple Model Adaptive Estimation (MMAE), Multiple Model Adaptive Control (MMAC), Distributed Kalman Filter (DKF), Fault Detection and Isolation (FDI), Jamming, Spoofing

178

15. SECURITY CLASSIFICATION OF REPORT

Unclassified

16. SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

17. SECURITY CLASSIFICATION OF ABSTRACT

Unclassified

18. SECURITY CLASSIFICATION OF ABSTRACT

UL